Towards Zero-Shot, Controllable Dialog Planning with LLMs

Dirk Väth[1], **Ngoc Thang Vu**[1],

[1]University of Stuttgart
dirk.vaeth@ims.uni-stuttgart.de, thang.vu@ims.uni-stuttgart.de

Abstract

Recently, Large Language Models (LLMs) have emerged as an alternative to training task-specific dialog agents, due to their broad reasoning capabilities and performance in zero-shot learning scenarios. However, many LLM-based dialog systems fall short in planning towards an overarching dialog goal and therefore cannot steer the conversation appropriately. Furthermore, these models struggle with hallucination, making them unsuitable for information access in sensitive domains, such as legal or medical domains, where correctness of information given to users is critical. The recently introduced task Conversational Tree Search (CTS) proposes the use of dialog graphs to avoid hallucination in sensitive domains, however, state-of-the-art agents are Reinforcement Learning (RL) based and require long training times, despite excelling at dialog strategy. This paper introduces a novel zero-shot method for controllable CTS agents, where LLMs guide the dialog planning through domain graphs by searching and pruning relevant graph nodes based on user interaction preferences. We show that these agents significantly outperform state-of-the-art CTS agents ($p < 0.0001$; Barnard Exact test) in simulation. This generalizes to all available CTS domains. Finally, we perform user evaluation to test the agent's performance in the wild, showing that our policy significantly ($p < 0.05$; Barnard Exact) improves task-success compared to the state-of-the-art RL-based CTS agent.

Code —
https://github.com/DigitalPhonetics/conversational-tree-search/tree/llm-policy

1 Introduction

Recently, LLMs have emerged as an alternative to training task-specific dialog agents. They excel in broad reasoning capabilities (Huang and Chang 2023), and demonstrate high performance in zero-shot learning scenarios (Wei et al. 2021; Brown 2020). However, LLMs-based dialog agents often fall short when planning towards a dialog goal is required. As the underlying LLMs are optimized towards following user instructions, such agents struggle to take initiative in proactively steering dialogs towards a fixed goal across multiple turns (Deng et al. 2024). Additionally, these models struggle with hallucination (Xu, Jain, and Kankanhalli 2024; Li et al. 2024), making them unsuitable for information access in sensitive domains where the information given to a user must be correct.

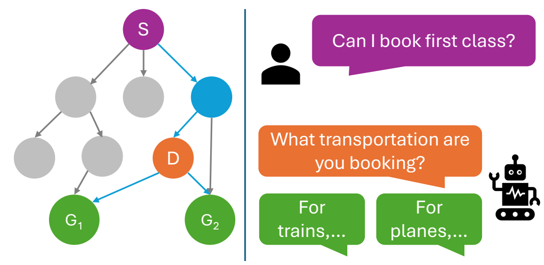

Figure 1: To plan a dialog, the proposed agent tracks possible user goals (green: G_1, G_2) based on the initial user input (purple: S), then plans paths (blue: $S \to \ldots \to D \to G_{1/2}$) to reach them, identifying decision points to proactively prompt the user for more information (orange: D).

Because of these challenges, common language-based interfaces for accessing information in sensitive domains rely on FAQ-style retrieval or hand-crafted dialog policies. However, while FAQ-style retrieval agents can deliver immediate answers, they are limited to general information not tailored to a users specific situation, which leads to either very general or overly long and complex responses. Furthermore, retrieval accuracy can suffer in large domains. Dialog systems with hand-crafted policies, on the other hand, can provide short and personalized responses, but are often perceived as frustratingly long and rigid.

The recently introduced task, CTS (Väth, Vanderlyn, and Vu 2023), proposes a graph-based framework for controllable task-oriented dialog in sensitive domains that interpolates between these two information seeking styles, implicitly adapting its behavior to match user interaction expectations. This flexibility allows CTS to combine the advantages of both settings, while avoiding their largest disadvantages. To prevent the policy from switching contexts or hallucinating false information, CTS requires a dialog agent to walk an expert-created dialog graph, deciding at each node whether it is relevant to the user and should be output, or which neighbor to skip to next. This graph walking approach requires the ability to plan efficient paths to possible goal nodes and to identify the decision points where additional user input is necessary. As RL-based algorithms are trained to optimize goal planning, Väth, Vanderlyn, and Vu (2023, 2024) propose RL-based agents for the CTS task.

Despite these advantages, a big drawback of the aforementioned RL-based state-of-the-art CTS agents is their long training time (up to 7 days on GPU for a graph with 123 nodes). This is particularly problematic as a new agent must be trained every time the graph is updated, even if only a single node changes, making it difficult to develop new dialog domains in an iterative way. Not only does this have a big impact on the environment and costs, but in sensitive domains such as legal domains, it could be unacceptable to have such long waiting times in cases where new laws or policies come into effect.

The goal of this paper is therefore to explore how to apply the reasoning and zero-shot capabilities of LLMs to dialog planning in a way that allows steering the conversation towards an overarching goal, while at the same time avoiding the hallucination problem and maintaining an expert-controllable dialog flow. To this end, we investigate the following research questions:

- **RQ1:** How can we optimize the runtime efficiency (to allow for real-time dialogs) and resource requirements of a zero-shot LLM-based CTS agent when planning dialog trajectories through a CTS domain graph?
- **RQ2:** How can we optimize the dialog success of such a zero-shot LLM-based CTS agent?
- **RQ3:** How does a zero-shot LLM-based approach to dialog planning compare to a trained CTS RL-agent in simulation?
- **RQ4:** How does a zero-shot LLM-based approach to dialog planning compare to a trained CTS RL-agent when testing with real users?

To address these questions, we investigate how LLMs can be leveraged to help plan conversations by first detecting the user preference for interaction style, searching the domain graph for relevant nodes, pruning the search results, and subsequently guiding the user across selected paths through the domain graph (see figure 1), while at the same time eliminating the need for training and adhering to the CTS controllability aspects. We validate our method against state-of-the-art RL-based agents in simulation on multiple domains. Finally, we evaluate real-world performance in a user study.

Our main contributions are: 1) We propose a novel, LLM-based method for a zero-shot, controllable task-oriented CTS agent: CTS-LLM. 2) We demonstrate strategies to improve the reasoning quality and inference speed of this LLM-based agent, making real-time dialog feasible. 3) We show that CTS-LLM is not only able to successfully plan and execute dialogs, but even improves dialog success in simulation compared to trained state-of-the-art RL-based CTS agents. These results generalize to all three available domains, when using large commercial as well as smaller scale local LLMs. 4) Finally, we demonstrate that improvements to dialog success translate to the real world through human evaluation on a single domain. All code and data are publicly available.

2 Related Work

In this section, we give a brief overview of the most closely related methods.

Dialog Planning

In task-oriented dialog systems, a policy, sometimes also called dialog manager, is responsible for selecting the dialog system's next actions. There are many different ways to implement dialog policies, such as using a fixed set of rules (Bobrow et al. 1977), statistical models (Kim et al. 2008), or even prompting or fine-tuning LLMs (Mi, Wang, and Li 2022).

One of the most frequently applied approaches is to train task-oriented dialog policies with RL because such agents naturally learn efficient planning strategies to reach user goals (Scheffler and Young 2002; Peng et al. 2018; Kwan et al. 2023). These RL agents are usually trained in an interactive setting, by performing dialogs against a simulated user (Schatzmann et al. 2006). Each dialog, the user simulator randomly selects a goal and then takes turns conversing with the RL agent, until either reaching this goal, or encountering another constraint such as maximum dialog length. While RL agents perform well in task-oriented dialog, a new dialog agent must be trained for every new dialog domain.

Conversational Tree Search

CTS (Väth, Vanderlyn, and Vu 2023) is a task-oriented, controllable dialog task which frames dialog planning as a graph search and traversal problem.

The graph, which describes the dialog domain, is structured by a human domain expert. Each node contains text that the dialog agent can output to the user, e.g. a greeting, information, or follow-up questions. As the possible output texts are defined by a human dialog designer, no hallucination is possible. Nodes are connected to neighbors via edges, where each edge is associated with a specific user intent, e.g., indicating interest in reimbursement or in booking. Every node is associated with a node type, such as question and variable nodes (for acquiring user input), logic nodes (choosing neighbor nodes based on variable values instead of user intents), and information nodes (information output without user input).

A CTS agent walks this graph node by node, starting from a unique greeting node. The goal is to identify and reach the node in the graph that best answers the user's information need. At each node, the agent makes the decision to either output the current node's text (e.g., asking a question or providing information), or to skip a neighboring node.

The task supports different user interaction preferences, accommodating both users with a vague or general information need, as well as users with more concrete questions. Based on the predicted interaction style, an agent can either traverse the graph by 1) follow a rigid scheme where the system outputs the text of each node visited, e.g., if the user information need is very vague, 2) skip directly to an answer, e.g., if the user question is very concrete, or 3) employ a hybrid approach, asking only questions that are necessary to clarify the user's information need on the path to the nodes

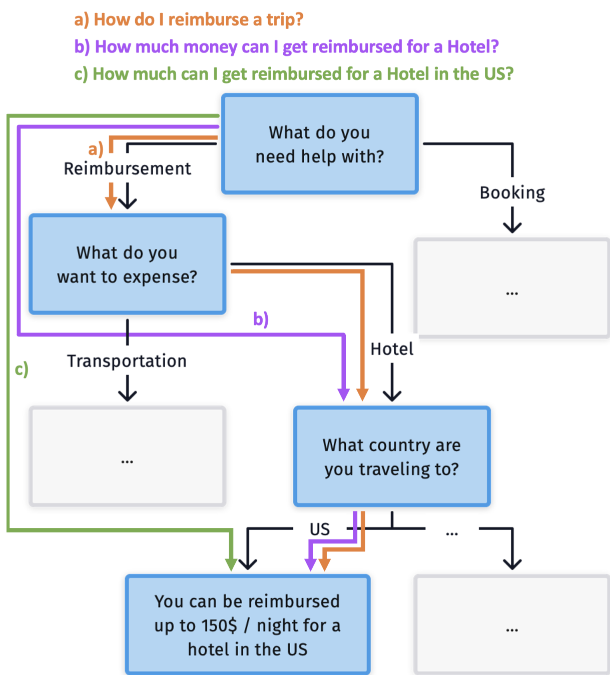

Figure 2: Example dialog graph and three different user inputs, leading to step-by-step conversation (a), clarification steps (b), or directly skipping to the answer (c) (Väth, Vanderlyn, and Vu 2023)

that are candidates for potentially providing the requested information. Figure 2 shows an example of three different user inputs, and the three different resulting interactions. As the agent can only output predefined text or filled templates, the output information is controllable by the dialog designer. Regardless of interactions style, the agent must walk the dialog tree node by node, meaning it cannot suddenly switch contexts.

Current state-of-the art CTS agents are RL-based (Väth, Vanderlyn, and Vu 2023, 2024), and while demonstrating good task performance, require long training times if any portion of the graph is changed. To this end, our goal is to replace the RL-based agent with one that can zero-shot adapt to any graph changes or even new graphs.

Zero-Shot Learning

Recent research into LLMs has demonstrated their strong abilities as zero-shot learners (Gruver et al. 2023). This performance can be improved by fine-tuning models on tasks which are described by textual instructions (Wei et al. 2021). Brown (2020) further show that similar improvements can be obtained by providing in-context examples. Rather than providing only instructions for the task the model should accomplish, the authors additionally included examples of how to do the task in their prompt. Furthermore, Schick and Schütze (2021) found that in-context examples are not just useful for very large LLMs, but can also be used to improve the performance of smaller, less resource intensive language models.

Information Retrieval

As this paper focuses on information seeking scenarios, we also discuss related work in the realm of information retrieval. Although there has recently been a shift towards retrieval augmented generation (Lewis et al. 2020; Chen et al. 2024), these methods can still suffer from the same hallucination problems inherent to LLMs (Hersh 2024), making them unsuitable for sensitive domains. Therefore, we primarily discuss document based retrieval, which has long been an area of research (Mitra and Chaudhuri 2000; Ghorab et al. 2013; Wang et al. 2024). Early methods relied on cooccurrences of words or lemmas between the search query and the available documents (Salton 1983). Modern approaches tend to make use language embeddings that are better able to represent the meaning of documents (Wang et al. 2024), sometimes adding a filtering or re-ranking step to the retrieved documents (Zerveas et al. 2022). Our proposed method also makes use of such embedding models for the initial document retrieval, and an LLM for a filtering step.

Natural Language Understanding

Natural language understanding has been a long-standing goal in the NLP community (Schank 1972; Weld et al. 2022; Yu et al. 2023). Although many approaches have been applied over the years, from handcrafted rules (Seneff 1992), to probabilistic models (Kuhn and De Mori 1995; He and Young 2003), to deep learning (Xu and Sarikaya 2013), more recent approaches have begun leveraging the power of LLMs (Min et al. 2023). In this paradigm, rather than needing to learn a mapping from user utterances to a fixed set of slots and intents, the problem can be reformulated as a question answering task (Namazifar et al. 2021). Here, the goal is to leverage the knowledge and language understanding capabilities of language models to answer the question of "which intent/slot is the user trying to express?", and by doing so, to reduce or eliminate the need for training data.

Our method similarly uses LLMs to match user inputs against the set of possible intents (edges) at a given dialog node, following the most similar intent (edge).

3 Method

Our proposed LLM dialog system is a module based system (see figure 3) with the following components: 1) a dialog policy, 2) a Natural Language Understanding (NLU) unit, and 3) a dialog state tracker, storing variable values to fill templates. The dialog policy consists of a) an LLM-based interaction mode classifier, which decides whether the user inputs a statement or a question, b) an LLM-based intent classifier, to determine per node which intent, i.e., edge, a user was trying to select, c) an LLM-based filter module, searching node candidates in the dialog graph that might answer a user's question, and d) a navigation module responsible for walking the dialog graph and choosing to output or skip a node,

The policy adheres to the CTS task by traversing the dialog graph node by node, only outputting predefined node texts, thus remaining controllable. Even though we use

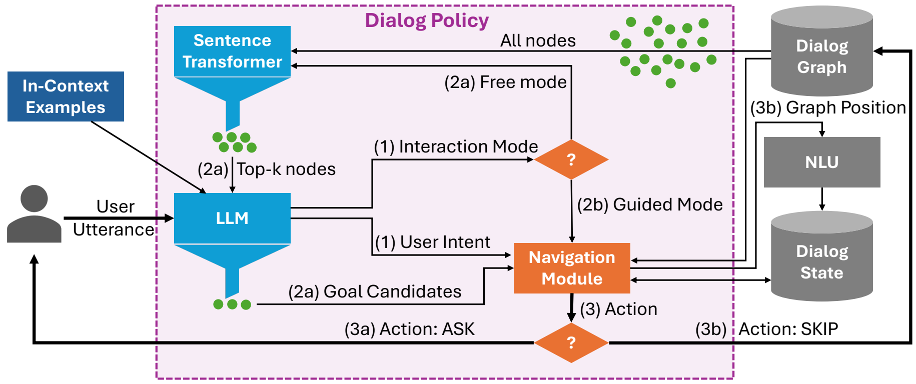

Figure 3: Interaction between user and system. The system predicts user intent and interaction mode from the user utterance (1). If the interaction mode is *free*, the user utterance is used to retrieve answer candidates from the dialog graph (2a). In *guided* mode (2b), the policy follows the edge associated with the predicted user intent (3b). The agent may also decide to ask for clarification or output information (3a).

LLMs to make decisions in how to traverse the graph, their outputs are never shown to the user.

Dialog Policy

In contrast to RL-based agents, the dialog planning in our method is performed by the combination of an LLM and graph algorithms. Concretely, the dialog policy uses the same LLM for the components a), b) and c) to coordinate planning (see algorithm 1 for a concise overview).

Navigation Module The policy initiates a dialog by outputting the start node of the dialog graph to the user. After receiving the first input from the user, the interaction mode classifier (step 1 in figure 3) is called to decide whether the user expresses a vague information need, or poses a concrete question, then proceeding with either step:

(2a) If the user input was classified as a vague or general information need, the following dialog interaction is considered a "guided" turn-by-turn mode with the dialog system in the lead, supporting users unfamiliar with the domain or with a vague information need. That is, the policy will always output the content of the current node to the user and, if necessary, wait for user input to continue (step 3a in figure 3). The user's input will be matched against the possible intents available at that node (step 2a in figure 3). Then, the policy will move to the corresponding neighboring node (step 3b in figure 3). Dialog continues in this manner until the tree end is reached or the user ends the dialog.

(2b) If the user input was classified as a (concrete) question, the following dialog interaction is considered a "free" mode, with the goal to answer the user's question as efficiently as possible, only prompting the user when disambiguation is required to clarify the exact information need. From the dialog graph $G = (V, E)$, with nodes V and edges E, a list of possible goal node candidates $V_g \subseteq V$ is retrieved by the filter module (step 2b in figure 3). A node is considered a goal candidate if it might answer the user question. For each goal node candidate $v_g \in V_g$, the policy calculates all possible paths $P_g := \{(v_1, \ldots, v_n) | v_n = v_g \wedge \forall v_i, v_{i+1} : (v_i, v_{i+1}) \in E\}$ in the dialog graph that lead from the current node $v_1 \in V$ to that goal node. Between the list of paths to each goal node, it selects the longest shared path prefix reaching all goal nodes. Here, we calculate the shared prefixes by looking at all paths to the candidate goals: $P_{shared} := \{(v_1, \ldots, v_n) | \forall v_g \in V_g \exists p \in P_g \forall 1 \leq i \leq n : p_i = v_i\}$. Once the longest prefix $argmax_{p \in P_{shared}} |p|$ has been identified, the policy walks through this prefix without outputting any of the nodes to the user until it reaches the final node (see step 3b in figure 3). The final node in the prefix is then output to the user (step 3a in figure 3), because it is by definition either a decision point where user input is required, or a goal node itself. If there are still open goals, the user intent is identified, the policy moves to the corresponding neighbor, and a new longest prefix is calculated using the remaining goal nodes. We use the longest shared path prefix as users in our pilot study perceived this approach as more usable, e.g., compared to using shortest paths, as the decisions at the tail of the paths seem more related to the user question (closer to the goal) than at the beginning of the paths, where topics might still be quite general (farther removed from the goal).

As variable questions do not cause splits in the graph, they are skipped. Whenever the policy reaches a node containing a template, but does not know the necessary values from the Dialog State Tracker (DST) yet, it looks back through the history of visited nodes until if finds the corresponding variable node. The policy then requests the variable value from the user, updates the beliefstate, and continues the dialog from the template node.

The full planning process is visualized in figure 1, where the policy calculates paths (blue) to two possible goal candidates (green), asking for clarification only at the orange node (last node on longest shared path prefix) which is directly connected to the two possible goal candidates, rather than at preceding branching points (blue).

Interaction Mode Classification Interaction mode classification is the first step in planning dialogs with our proposed method. We classify the user utterance into the classes

Algorithm 1: Dialog Policy

```
Data: graph, input, node
done ← false ;
mode ← mode_classifier(input);
if mode = QUESTION then
    goals ← goal_filter(graph, input);
    path ← longest_prefix(goals);
end
while done ≠ true do
    if mode = QUESTION then
        node ← last_node(path);
    end
    output(node);
    if node.type ∈ {QUESTION, VARIABLE} then
        input ← input();
        next_node ←
          neighbors(node)[intent(input)];
        path ← longest_prefix(goals);
    else
        next_node ← neighbor(node);
    end
    if node ∈ goals then
        goals ← goals \ {node};
    end
    node ← next_node;
    done ← goals = ∅ ∨ neighbors(node) = ∅;
end
```

guided or *free* by prompting an LLM to determine if a given user input is a question/command/request or not (see appendix B for the exact prompt). We then parse the generated outputs *'yes', 'command', 'request'* as *free* and all other outputs as *guided*. Using an LLM for this task avoids training a specialized classifier and detects the interaction mode better than, e.g., off-the-shelf question classifiers, as many user inputs that are asking for information might not be phrased as questions.

Intent Classification Each decision (question) node has a list of possible user intents. To decide what intent the user is trying to express, we prompt an LLM with the user utterance and a JSON list containing the intent candidates and their indices. We ask the model to output the index of the response that best matches the user's input (see appendix B). We then extract this index and move to the corresponding neighbor.

Goal Node Filter The goal node filtering step is responsible for the path selection aspect of dialog planning in *free* policy mode. We perform a similarity ranking between the user utterance and the list of all nodes in the graph using a fast semantic search model, retrieving the 15 most similar nodes as a pre-filtering step. These nodes are then passed into an LLM to reason about which candidates could actually be answers to the user's question. Performing the retrieval step with a much smaller model instead of giving the full list of nodes to the LLM keeps the number of input nodes to the LLM constant, regardless of graph size. This reduces the number of input tokens and the search space to be reasoned over, thus simplifying the problem and reducing computation time, while also increasing the accuracy of the results. However, fast retrieval methods usually obtain reasonable recall scores for $k >> 1$, and thus the top-1 result cannot be trusted (also demonstrated in figure 4).

We therefore apply LLM-based reasoning to the 15 answer node candidates conditioned on user utterance to filter out the most relevant candidates regardless of their retrieval rank. We then ask the model to output all nodes which answer the question, along with a justification for why each node does so. To improve the reasoning process, we add out-of-domain, in-context examples that represent positive and negative question/answer pairs. For example, we add: '*What is the weather usually in Singapore at 9 a.m.?*', and as a positive example fact: '*In Singapore, between 8 a.m. and 11 a.m., the weather is around 35 degrees Celsius.*' with the justification: '*The fact is relevant as it answers the user request, and the requested time of 9 a.m. lies between the fact's timespan of 8 a.m. to 11 a.m.*'. The full prompt can be seen in appendix B.

4 Datasets

To investigate how our method compares to state-of-the-art RL-based CTS agents, we evaluate on the three available English datasets: REIMBURSE-en, DIAGNOSE and ONBOARD. REIMBURSE-en contains information about travel reimbursement procedures, DIAGNOSE is a medical help domain, and ONBOARD provides support for foreigners moving to a new city.

Each of these datasets contains a dialog graph, user questions and user responses. User questions are associated with the dialog node that provides an answer. User responses are paraphrases for the possible user intents at each node. Statistics for each dataset can be found in table 1.

5 User Study

To assess the performance of our proposed CTS-LLM agent, we performed user evaluation. To be able to compare our results to Väth, Vanderlyn, and Vu (2024), we performed a parallel study adhering to the same design, methodology, and REIMBURSE-en domain. Although we only evaluated the CTS-LLM agent, we can thus treat the combination of both studies as a between-subject design.

Study Design

All participants were recruited via the crowdsourcing platform Prolific and paid at a rate of 12.60€/hr, consistent with minimum wage in the country of our research institution. They were asked to provide basic demographic information about their age, gender, and experience with chatbots and business travel. Following Väth, Vanderlyn, and Vu (2024), subjects were then asked to conduct three conversations with the chatbot in the REIMBURSE-en domain. For each dialog they were randomly assigned an information goal they should try to reach. These goals represented 1) an open question, e.g., the general process of how to book a hotel, or what to do in an emergency, 2) a simple, concrete question, e.g.,

Dataset	Split	#Nodes	Tree Depth	Max. Node Degree	#User Questions	Avg. User Questions	#Answer Paraphrases	Avg. Answer Paraphrases
REIMBURSE-en	Train	123	32	14	279	3.5	246	3.4
REIMBURSE-en	Test	123	32	14	173	2.2	162	2.2
DIAGNOSE	Test	98	10	6	150	2.0	298	3.0
ONBOARD	Test	88	15	9	117	2.0	152	2.7

Table 1: Overview of the *REIMBURSE-en*, *ONBOARD* and *DIAGNOSE* datasets (Väth, Vanderlyn, and Vu 2024).

if they could get reimbursed for seat reservations on a train, or 3) a complex, concrete question, e.g., how much money they could be reimbursed per day if they planned to stay at their brother's apartment on their upcoming trip to France.

After each dialog, users were asked to rate the subjective quality and length, using two single-item Likert questions. Subjective length was rated on a 5-point scale from 1 (much too short) to 5 (much too long). Success was rated on a 4-point scale from 1 ('my question was not answered at all') to 4 ('my question was entirely answered'). After the final dialog, participants were also asked to provide free-form feedback on their perception of the agent as well Likert feedback on the usability of the agent using the UMUX scale (Finstad 2010). They were also asked about their trust in the agent and perceived reliability using the Trust and Reliability subscales of Körber (2018)'s Trust in Automation scale.

Pilot Study

Before conducting the main user study, we performed a pilot study with 12 users to check the technical setup. Here, we found that users were quickly frustrated when they had to answer multiple questions at the beginning of the dialog tree, as they took this to mean that the agent had not understood their question. This motivated our design decision to delay clarifications by calculating the longest shared path prefix instead of shortest paths.

Main Study

For the main study, we recruited 24 participants. Of these, 13 identified as female and 11 as male, with ages ranging from under 20 to 59 years old. On average, participates had decent familiarity with chatbots (3.5 on a 5-point Likert scale) and some familiarity with business trips (2.5 on a 5-point Likert scale). Each user participated in 3 dialogs, resulting in a total of 72 collected dialogs. After removing 4 dialogs due to technical errors or bad-faith participation, we were left with 68 total dialogs. Combining these with the 61 dialogs collected for CTS-RL by the parallel study from Väth, Vanderlyn, and Vu (2024) resulted in a total of 129 dialogs for the between-subject setup.

6 Results & Discussion

All results for the LLM-CTS agents were obtained using GPT-4o-mini (version 2024-07-18)[1] or the instruction tuned Gemma-2 9B (Team et al. 2024). We choose the multi-qa-mpnet-base-dot-v1 Sentence Transformer (Reimers and Gurevych 2019) for pre-filtering. Results for the RL-based CTS agents are taken from (Väth, Vanderlyn, and Vu 2024).

[1]https://platform.openai.com/docs/models/gpt-4o-mini

Objective Evaluation Metrics

To evaluate the objective performance, we follow Väth, Vanderlyn, and Vu (2024) and measure binary dialog success by evaluating whether the user's information need was fulfilled for each dialog. This means that the dialog agent has to reach the node containing the answer to the user's information goal as well as output the node text to the user. The reported values for dialog success measure the percentage of successful dialogs.

Dialog length refers to the number of dialog turns (agent turns + user turns) which are perceived by the user. Thus, dialog length only counts the number of user inputs and the number of system *ASK* actions. *SKIP* actions are not counted, as they are not visible to the user.

Finally, interaction mode classification is a binary classification measure (*free* or *guided*), measured once per dialog based on the prediction from the initial user utterance.

RQ1: Optimizing Resource Requirements & Runtime Efficiency

To save on hardware requirements, we try to solve as many of the policy sub-tasks as possible with the same LLM. We used Gemma-2 on the train split of the REIMBURSE-en dataset as a development setup.

After conducting preliminary experiments, we find that we can obtain good performance from the same LLM for interaction mode detection, user intent classification, and goal node candidate filtering. However, as can be seen from table 2, filtering goal node candidates directly from the full dialog graph proved too slow for real-time usage. In our experiments, we found that this approach to filtering takes $\sim$33.61 seconds, and is resource intensive because of the number of resulting input and output tokens (4601 and 158, respectively). This will only get worse as the graph size increases.

To address this, we add a small Sentence Transformer specialized in relevant passage retrieval (Reimers and Gurevych 2019) to pre-filter nodes based on semantic similarity to the user utterance. After performing a retrieval experiment on the REIMBURSE-en train split (see figure 4), we choose $k = 15$ as a good trade-off between candidate space reduction and accuracy.

Keeping all 15 goal candidate nodes, however, would result in the need for many clarifying questions and thus long dialogs. To mitigate this, we choose to apply the LLM as a post-filter to the 15 retrieved candidates. Together, the retrieval and post-filtering process only take $\sim$17.84 seconds,

Pre-filter	Justifications	In-Context Examples	Recall	Avg. Candidates per Question	Avg. time per Question (sec)	Avg. #tokens (input)	Avg. #tokens (output)	Avg. GPU memory (GB)
✓	✓	✓	**0.84**	2.37	17.84	1581	95	26.59
✓	✓	✗	0.35	0.91	25.85	935	147	26.49
✓	✗	✓	0.80	2.26	**6.66**	1346	31	26.56
✗	✓	✓	0.78	2.66	33.61	4601	158	27.04

Table 2: Recall and resource usage for the REIMBURSE-en train split with Gemma-2 on a NVIDIA A600 GPU for different goal candidate filtering processes.

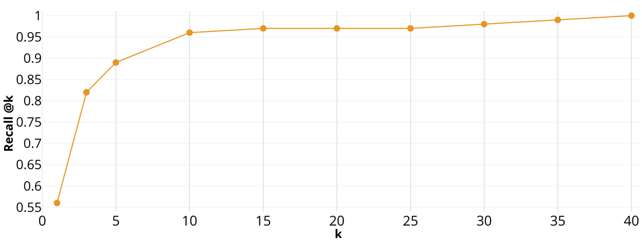

Figure 4: Recall of pre-filtering the possible goal nodes for different k on the REIMBURSE-en train split.

almost halving the execution time, making it feasible for real-time use. Since the retrieval step is very efficient, it scales better with graph size while the slower post-filter is now decoupled from the graph size. This allows for scaling to larger domains without a noticeable penalty on filter time. As we see in table 2, even though we add the additional retrieval model, the reduction in input (factor ~2.9) and output (factor ~1.7) tokens results in a lower total memory footprint: 26.59 GB, compared to 27.04 GB for the LLM without the retrieval step. Finally, the reduction in tokens and computation time also lowers the environmental impact, and especially for LLMs consumed via API, costs.

RQ2: Optimizing Dialog Success

Performance of *free* mode is fully dependent on the output quality of the filtering step, thus improving filter performance directly results in better dialog success. Again, we use Gemma-2 on the train split of the REIMBURSE-en dataset as a development setup.

As first tests with the two-stage filtering process did not yield good retrieval accuracy, we analyzed filtering failures. We observed, e.g., that some LLMs have trouble with numerical reasoning. For example, given the user question: '*How much money do I get for a 9 hour long business trip?*', the only relevant node returned by the LLM is '*For business trips lasting less than 8 hours, you are not entitled to a per diem*', which does not answer the question because the user's time requirement is outside the answer's time interval. This is particularly problematic, as there is another node that answers the question: '*For trips lasting more than 8 hours, but less than 14, you are entitled to a per diem of 6€.*'.

To improve the model performance and reasoning, we tested the following interventions 1) adding in-context examples to the model prompt and 2) adding an additional reasoning step by requiring the model to also output its justification for marking facts as relevant.

In the ablation study in table 2, we can see that both the in-context examples and additional reasoning step result in improved recall. While justifications slow the model down by generating more output tokens, we consider the improvement in recall as more important, as retrieval errors define the upper bound on task performance.

Based on these experiments, we see that combining pre-filtering, justifications and in-context examples provides the best recall, and a good trade-off in response time and resource usage.

RQ3: Performance In Simulation

Evaluating dialog performance against a simulated user is common practice in the dialog community (Ai and Weng 2008), where it is expensive to test development systems on real users. To this end, the CTS task provides a user simulator to evaluate the system's performance (Väth, Vanderlyn, and Vu 2023). To simulate a dialog, a random goal is chosen from the dialog graph along with a tree path to reach that goal. The interaction mode is set to either *free* mode or *guided* mode. When free mode is chosen, a random user question associated with the goal node is drawn as the first user utterance. If guided mode is chosen, the simulator will select a paraphrase of one of the pre-defined intents at the start node as first utterance. For each node along the goal path that accepts user input, a random paraphrase of that intent will be chosen . If the simulated user reaches their goal, the dialog is rated successful (see Appendix C for more details). Table 3 shows performance metrics against the CTS simulator when evaluating on 500 simulated dialogs for our LLM agent and a state-of-the-art RL agent (Väth, Vanderlyn, and Vu 2024) on the test splits of all three domains.

Using GPT-4o-mini, our method significantly improves dialog success in all three domains ($p < 0.0001$; Barnard Exact test). From this, we see that in addition to the sustainability advantages our proposed agent brings by eliminating the need for training, it is also able to outperform the previous RL-based state-of-the-art CTS agent. We also see that this performance generalizes across domains.

Additionally, when looking at the results for the Gemma-based agent, we find that even a smaller LLM, which can be run locally (e.g. on a single NVIDIA A600), can improve dialog success on all domains compared to the RL-based CTS agent, with significant improvements in the ONBOARD and DIAGNOSE domains ($p < 0.001$; Barnard Exact test).

Domain	Dialog Policy	Dialog Success (%)	Dialog Length (Guided)	Dialog Length (Free)	Interaction Mode (F1)
REIMBURSE-en	CTS-RL	73.86	13.56	2.95	0.94
	CTS-LLM (Gemma-2)	77.00	10.79	2.91	0.89
	CTS-LLM (GPT-4o-mini)	**84.20****	10.31	**2.79**	**0.95**
DIAGNOSE	CTS-RL	76.31	6.42	**2.29**	n/a
	CTS-LLM (Gemma-2)	94.60**	5.24	3.36	**1.0**
	CTS-LLM (GPT-4o-mini)	**98.80****	5.26	3.52	**1.0**
ONBOARD	CTS-RL	73.61	7.88	2.98	n/a
	CTS-LLM (Gemma-2)	95.00**	4.94	**2.96**	**1.0**
	CTS-LLM (GPT-4o-mini)	**96.00****	4.88	**2.96**	0.99

Table 3: Objective performance metrics of our proposed CTS-LLM agent compared to the RL-based CTS baseline (Väth, Vanderlyn, and Vu 2024) ** represents significant increases ($p < 0.0001$ Barnard Exact test). We bold the best results per model and domain, except for guided dialog length, where the goal is to obtain context, rather than simply efficient navigation.

RQ4: Performance With Real Users

For the user evaluation, we selected the GPT-4o-mini-based policy, as it achieved the highest task success in simulation. To compare it with a state-of-the-art RL-based CTS agent, we use the results reported for the REIMBURSE-en domain in Väth, Vanderlyn, and Vu (2024) and conduct a parallel user study with our LLM-based agent. As no user studies exist for the RL agent on other domains, recruiting sufficient participants to evaluate both agents across multiple domains would have been prohibitively expensive. Thus, we focus on the challenging real-world REIMBURSE-en domain in this study. Example interactions can be found in Appendix D.

Policy	Success (%)	Avg. Turns	Subj. Length	Subj. Quality
CTS-RL	77.05	**7.38**	**2.92**	2.87
CTS-LLM	**86.76***	10.24	2.75	**2.94**

Table 4: User study results for objective and subjective dialog metrics for CTS-RL (Väth, Vanderlyn, and Vu 2024) and our method. * represents significant differences ($p < 0.05$). Perceived quality is a 4-point scale (1 = not answered; 4 = fully answered), perceived length a 5-point scale (1 = much too short; 5 = much too long).

Table 4 shows the results of our policy compared to the results of the RL-based CTS agent. Performing a Barnard-Exact test, we find that our method led to a significant increase in dialog success compared to the RL-based CTS agent ($p < 0.05$) without causing any negative effects in how users perceived the quality of answers they received. We additionally find that although the CTS-LLM dialogs were objectively longer, the subjective length of the system was rated similarly (even slightly shorter) to the RL-based CTS agent. The shorter paths from the RL agent are likely explained by the optimization goal, which optimizes dialog length. Thus, a future goal is to improve the dialog length in the planning step.

When looking at the subjective perception of the agent in table 5, we find that users rated both reliability and usability of our proposed method higher.

Policy	Trust	Reliability	Usability
CTS-RL	**3.16**	2.96	62.83
CTS-LLM	3.00	**3.06**	**67.90**

Table 5: User study results for perceived trust, reliability and usability. No significant differences were found between CTS-RL (Väth, Vanderlyn, and Vu 2024) and our results.

7 Conclusion

In this paper, we explore how to apply the reasoning and zero-shot capabilities of LLMs to dialog planning in a way that allows steering the conversation towards an overarching goal, while at the same time avoiding hallucination and retaining an expert-controllable dialog flow. We introduce a novel task-oriented dialog policy that performs dialog planning based on a combination of a fast Sentence Transformer, LLMs, and graph algorithms. Our dialog policy conforms to the controllability aspect of the CTS task by using LLMs to help with path planning, rather than generating output for the user, avoiding the possibility of hallucination. This approach also removes the need for training CTS dialog policies, allowing for instantaneous adaptation to changes in domain graphs.

Although RL is known for its planning capabilities, especially in the task-oriented dialog policy setting, we demonstrate that our LLM-based approach to dialog planning can significantly outperform state-of-the art RL-based dialog policies. These results hold across all three domains tested in simulation, and also translate to evaluation on the REIMBURSE-en domain with real users.

In addition, we identify computational resource and reasoning challenges, and demonstrate working solutions to speed up response time and reduce the number of input and output tokens. In this way, we improve policy performance and effectively decouple the slow filtering step from the graph size.

Finally, our simulations demonstrate that our approach yields good results not only using very large commercial LLM, but also when using a smaller open-source LLM that can be run on a single GPU. This allows for more data safety and less costly experimentation and deployment.

A Limitations

Although we list the exact version numbers of the LLMs we use, commercial LLMs might get updated, thus impacting the exact reproducibility of our results obtained using commercial LLMs.

Additionally, at the time of writing, there are only three domains available for the CTS task. Therefore, we can evaluate our proposed method only on these domains. The evaluation with real users is limited to a single domain due to cost reasons.

As discussed in the results, the LLM policy interactions, although perceived shorter than the RL policy interactions in the user study, are objectively several turns longer than the RL policy interactions.

Although we save the computational resources required to train CTS-RL agents, the runtime resources of the CTS-LLM are higher.

B LLM Prompts

Interaction Mode Classification

Table 6 shows the full prompt for the interaction mode classifier.

Role	Prompt
variables	*user_utterance*
system	Answer with "yes" or "no" only.
user	Is the following text a question / command / requesting or not: *user_utterance*?

Table 6: Prompt for classifying which interaction mode (guided or free) the user wants.

Intent Classification

Table 7 shows the full prompt for the user intent classifier.

Role	Prompt
variables	*user_utterance*, *intent_candidates*
system	Given this list of possible response candidates: *intent_candidates* Decide which of the response candidate texts most closely matches the user intent, and only output the responses index. Do not output any other text, any code, or anything else.
user	*user_utterance*

Table 7: Prompt for classifying which pre-given intent the user is most likely trying to choose.

Goal Node Filter

Table 8 shows the full prompt for the goal node filtering step. The variable *examples* is filled with the following text:

For example, given the facts:

Role	Prompt
variables	*user_utterance*, *node_texts*, *examples*
system	You will be provided with a json list of facts and a query. You are to act as a first filter to decide which of the given facts answer the query or are relevant to answering the query, at least partially, and which ones are not relevant to answering the query at all? Assign each fact a relevance indicator between 0 and 2, and add a justification of why it is relevant (2), partially related (1), or irrelevant (0). Facts are also considered relevant if they imply the answer. If facts contain placeholders inside curly braces, assume the placeholder will be filled with a reasonable value. Don't return anything besides the json list of relevant facts, and only return facts with relevance indicator higher than 0. Don't return code or additional text. REMEMBER: even if some facts are only slightly relevant to answering the query, it is better to rate them with a relevance of 1 than to have all facts have relevance 0. *examples* Note that the fact with key 4 was excluded from the output, as it has a relevance of 0: Fact 4 is not related to the query about the weather in Singapore.
user	======= Facts ======= *node_texts* ======= Query ======= *user_utterance*

Table 8: Prompt for filtering to keep only relevant goal node candidates

```
[{"key": 0, "fact": "In Singapore,
   at 9 a.m., it is usually around
   35 degrees celsius."},
{"key": 1, "fact": "In Singapore,
   between 8 a.m. and 11 a.m., the
   weather is around 35 degrees
   celsius."},
{"key": 2, "fact": "In London, at 9
    a.m., it is usually 25 degrees
   celsius."},
{"key": 3, "fact": "In Singapore,
   between 10 a.m. and 11 a.m., it
   is usually around 30 degrees
   celsius."},
{"key": 4, "fact": "In Singapore,
   there are many tourist
   attractions."},
{"key": 5, "fact": In Singapore, it
    is usually around 35 degrees
   celsius in the mornings, but
   cooler in the evenings."},
{"key": 6, "fact": "In {{ COUNTRY
```

```
}}, at 9 a.m., it is usually
around 35 degrees celsius."}]
```

And a query: "What is the weather usually in Singapore at 9 a.m.?" The reply should only be a json list of the facts, indicating if the facts are related to or directly answering the query, formatted like this:

```
[{"key": 0, "relevance": 2, "
   justification": "The fact is
   relevant because it answers the
   user request perfectly"},
{"key": 1, "relevance": 2, "
   justification": "The fact is
   relevant as it answers the user
   request, because the requested
   time of 9 a.m. lies between the
   fact's timespan of 8 a.m. t0 11
   a.m."},
{"key": 2, "relevance": 1, "
   justification": "While the time
   is correct, the fact is listing
   the temperature for London
   instead of Singapore"},
{"key": 3, "relevance": 1, "
   justification": "The fact is
   talking about the weather in
   Singapore, which is relevant to
   the user, although the requested
    time of 9 a.m. lies outside the
    fact's timespan of 10 a.m. to
   11 a.m."},
{"key": 5, "relevance": 2, "
   justification": "The fact is
   relevant as it partially answers
    the user query: while it does
   not state a specific time, it
   implies the temperatures in
   Singapore at the requested time
   "},
{"key": 6, "relevance": 2, "
   justification": "The fact is
   relevant as it could answer the
   user request perfectly, once the
    placeholder is filled."}]
```

C User Simulator

All dialog success scores and conversation lengths reported from simulated users are the average across 500 dialogs.

At the start of each simulated dialog, a goal node is drawn randomly from the domain graph. Each interaction then begins at the (unique) start node, which is output to the user. Here, the initial user utterance is generated depending on the dialog mode. In free mode, a question associated with the goal node is drawn randomly. In guided mode, a paraphrase for the user intent on the path to the goal is randomly chosen instead. Control is then handed over to the policy. When the policy chooses to output a node to the user that requires a user response, the user response is randomly chosen from a list of available paraphrases associated with a user intent that facilitates reaching the goal. A dialog ends once the goal was reached, the user patience was reached (encountering the same dialog node multiple times), or the maximum turn length was reached (four times the maximum tree depth). To support the training of RL agents, each turn is stored in an experience buffer. The simplified pseudo-code for the user simulator is shown in listing 2. For more details, we refer to Väth, Vanderlyn, and Vu (2023) and Väth, Vanderlyn, and Vu (2024).

Algorithm 2: Pseudo code for the user simulator, adapted from Väth, Vanderlyn, and Vu (2023) and Väth, Vanderlyn, and Vu (2024).

Data: questions $\leftarrow$
  List of Questions associated with dialog tree nodes
Data: answers $\leftarrow$ List of Answer synonyms
Data: $N \leftarrow$ Number of dialogs to be simulated
Data: $G(V, E)$; // Dialog Tree
Data: $T \leftarrow$ Maximum number of turns per dialog
$n \leftarrow 0$;
for $n < N$ **do**
  $v \leftarrow$ start node;
  $m \leftarrow random\_uniform(\{\text{Free}, \text{Guided}\})$;
  $g \leftarrow random\_uniform(\{g \in V : |questions(g)| \geq 1\})$;
  $p \leftarrow (e(v, v_1), \ldots, e(v_N, g)) \subset E$;
  $u \leftarrow random\_uniform(questions(g))$;
  $t \leftarrow 1$;
  while $v \neq g \wedge t \leq T \wedge neighbors(v) \neq \emptyset$ **do**
    $a \leftarrow policy(v, u)$; // Select action
    if $a = ASK(v)$ **then**
      $print(v)$;
      $e \leftarrow e(v, v')$; // v' is g (guided mode) or next node after v (free mode)
      if $type(v) = information$ **then**
        $u \leftarrow \emptyset$;
      else
        $u \leftarrow random\_uniform(questions(e))$
      end
    else
      if $a = SKIP(v,v')$ **then**
        $v \leftarrow v'$;
        $u \leftarrow \emptyset$;
      end
    end
    $r \leftarrow$ Calculate rewards;
    Store a, v', r in replay buffer;
    $t \leftarrow t + 1$;
  end
  $n \leftarrow n + 1$;
end

D Example Interactions

In this section we provide examples of different types of interactions users had with our LLM-based CTS agent.

In figure 5, we see an example of a user who had a concrete question and was given a concrete answer. In this case, our agent found two possible goals along the same branch in the dialog graph that might answer the user's query, as there were no branches between the two answers which could determine which was more relevant, both were output, leading to a more complete answer being given to the user.

GOAL NODE: (16460328708250870): If you are also transporting other coworkers, you are entitled to 2 Cent per kilometer.

USER INSTRUCTIONS: You want to figure out how much money you can get for your travel.
1. You used your own car.
2. You took two colleagues with you.

SYSTEM: (NODE: 16348058621438633) What topic do you have questions about? You can either click on an answer from the suggested topics or enter your own text.

USER: Hi, on my recent trip I took my own car and carried two colleagues, how much money can I claim back for this please?

SYSTEM: (NODE: 16460325754790153) If you used your car for a pressing reason, you are entitled to 0.25€ per kilometer driven.

SYSTEM: (NODE: 16460328708250870) If you are also transporting other coworkers, you are entitled to 2 Cent per person and kilometer for each passenger who is also eligible for travel reimbursement. (To claim a passenger, they must also sign your travel reimbursement request.) If you are transporting business-related items (e.g., files) weighing more than 50 Kg, you are eligible for 2 Cent per 50 Kg and kilometer. (Changes coming in 2022)

DIALOG END: SUCCESS
SUBJECTIVE QUALITY: 3
SUBJECTIVE LENGTH: 2

Figure 5: Example dialog of a direct question which is given a direct answer. Excerpt from the conversations with user *c7bb706cdb623739bf9e80c638e4eb*.

In figure 6, we again see a user asking a direct question, however in this case, the system has two possible answer candidates (train or plane seat reservations) and needs to ask the user to clarify which of these is correct in order to answer. Once the user does this, there is only one answer candidate left and this is directly output to the user.

GOAL NODE: (16363756478730906): Seat reservations are allowed for train travel.

USER INSTRUCTIONS: You want to know if you can get reimbursed if you reserve a seat for yourself on the train.

SYSTEM: (NODE: 16348058621438633) What topic do you have questions about? You can either click on an answer from the suggested topics or enter your own text.

USER: I want to book a seat on the train. Can I get a refund if needed?

SYSTEM: (NODE: 16348100109577394) What type of transportation would you like?

USER: Train

SYSTEM: (NODE: 16363756478730906) Seat reservations are allowed for train travel.

DIALOG END: SUCCESS
SUBJECTIVE QUALITY: 2
SUBJECTIVE LENGTH: 3

Figure 6: Example dialog of a direct question which needs clarification before being answered. Excerpt from the conversations with user *6248a541d241dda06d678a24a511ce*.

In figure 7, we see a different type of dialog interaction. Here the user does not formulate a direct question and instead chooses one of the offered general topics. From here, the agent guides the user through each node in that branch of the dialog tree, giving them additional context and asking clarifying questions until they get the answer to their question.

GOAL NODE: (16370483534787100): Your business trip must be planned as though you were traveling alone.

USER INSTRUCTIONS: : You want to know how the reimbursement process works for a research semester.
1. You plan to bring your family with you.

SYSTEM: (NODE: 16348058621438633) What topic do you have questions about? You can either click on an answer from the suggested topics or enter your own text.

USER: Research semester

SYSTEM: (NODE: 16370487978253782) How long will your research semester last?

USER: 2 days

SYSTEM: (NODE: 16353297472197738) Is this as part of a reallocation?

USER: No

SYSTEM: (NODE: 16370501863636503) As there has been no reallocation, your research semester counts as a business trip.

SYSTEM: (NODE: 16353303012891903) Are you traveling alone or with your family?

USER: With family

SYSTEM: (NODE: 16370483534787100) Your business trip must be planned as though you were traveling alone (without your family).
How does that work? 1. Provide receipts for only the costs associated with purchases for yourself. 2. Collect comparison offers where necessary (e.g., transportation costs and accommodation) for what the costs would have been for only one person. 3. Personal expenses must be strictly separated from business expenses.

You are responsible for paying all additional costs that result from bringing your family along.

DIALOG END: SUCCESS
SUBJECTIVE QUALITY: 1
SUBJECTIVE LENGTH: 2

Figure 7: Example dialog of a guided mode interaction, where the user does not ask a direct question and must be asked follow-up questions to clarify their intent. Excerpt from the conversations with user *c017850ef3e9fd55350d0c8374859f*.

Ethical Considerations and Risks

To ensure that users could give informed consent, we provided a detailed description of the task and research objectives both on the crowdsourcing platform and once they had accepted the task. In respect of participant privacy, we specifically did not collect personally identifying data from any users. To this end, we store all logs and survey responses using an anonymous hash generated based on a given username, rather than with the username itself. In this way, users could log in again if they needed to take a break in the middle of the interaction, but we had no way of directly linking any recorded results to, e.g., users' Prolific account identifiers. To ensure that participants were fairly compensated, we followed best practices recommended by the crowdsourcing platform paying users at 12.60€/hr, which was in-line with minimum wage in the country of our research institution. We additionally used our pilot study to verify that our estimated time was below the median time we selected when advertising the task, meaning most participants had a higher hourly wage.

In terms of risks, the goal of this paper is to create more effective dialog agents with zero training effort. While we try to mitigate the possibility of chatbots generating false information, our work has the possible risk of creating chatbots that could also be used to replace human jobs.

References

Ai, H.; and Weng, F. 2008. User Simulation as Testing for Spoken Dialog Systems. In Schlangen, D.; and Hockey, B. A., eds., *Proceedings of the 9th SIGdial Workshop on Discourse and Dialogue*, 164–171. Columbus, Ohio: Association for Computational Linguistics.

Bobrow, D. G.; Kaplan, R. M.; Kay, M.; Norman, D. A.; Thompson, H.; and Winograd, T. 1977. GUS, a frame-driven dialog system. *Artif. Intell.*, 8(2): 155–173.

Brown, T. B. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Chen, J.; Lin, H.; Han, X.; and Sun, L. 2024. Benchmarking Large Language Models in Retrieval-Augmented Generation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16): 17754–17762.

Deng, Y.; Zhang, W.; Lam, W.; Ng, S.-K.; and Chua, T.-S. 2024. Plug-and-Play Policy Planner for Large Language Model Powered Dialogue Agents. In *The Twelfth International Conference on Learning Representations*.

Finstad, K. 2010. The usability metric for user experience. *Interacting with computers*, 22(5): 323–327.

Ghorab, M. R.; Zhou, D.; O'Connor, A.; and Wade, V. 2013. Personalised Information Retrieval: survey and classification. *User Modeling and User-Adapted Interaction*, 23(4): 381–443.

Gruver, N.; Finzi, M.; Qiu, S.; and Wilson, A. G. 2023. Large Language Models Are Zero-Shot Time Series Forecasters. In Oh, A.; Naumann, T.; Globerson, A.; Saenko, K.; Hardt, M.; and Levine, S., eds., *Advances in Neural Information Processing Systems*, volume 36, 19622–19635. Curran Associates, Inc.

He, Y.; and Young, S. 2003. A data-driven spoken language understanding system. In *2003 IEEE Workshop on Automatic Speech Recognition and Understanding (IEEE Cat. No.03EX721)*, 583–588.

Hersh, W. 2024. Search still matters: information retrieval in the era of generative AI. *Journal of the American Medical Informatics Association*, 31(9): 2159–2161.

Huang, J.; and Chang, K. C.-C. 2023. Towards Reasoning in Large Language Models: A Survey. In Rogers, A.; Boyd-Graber, J.; and Okazaki, N., eds., *Findings of the Association for Computational Linguistics: ACL 2023*, 1049–1065. Toronto, Canada: Association for Computational Linguistics.

Kim, K.; Lee, C.; Jung, S.; and Lee, G. G. 2008. A frame-based probabilistic framework for spoken dialog management using dialog examples. In *Proceedings of the 9th SIGdial Workshop on Discourse and Dialogue*, 120–127.

Körber, M. 2018. Theoretical considerations and development of a questionnaire to measure trust in automation. In *Congress of the International Ergonomics Association*, 13–30. Springer.

Kuhn, R.; and De Mori, R. 1995. The application of semantic classification trees to natural language understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(5): 449–460.

Kwan, W.-C.; Wang, H.-R.; Wang, H.-M.; and Wong, K.-F. 2023. A survey on recent advances and challenges in reinforcement learning methods for task-oriented dialogue policy learning. *Machine Intelligence Research*, 20(3): 318–334.

Lewis, P.; Perez, E.; Piktus, A.; Petroni, F.; Karpukhin, V.; Goyal, N.; Küttler, H.; Lewis, M.; Yih, W.-t.; Rocktäschel, T.; Riedel, S.; and Kiela, D. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; and Lin, H., eds., *Advances in Neural Information Processing Systems*, volume 33, 9459–9474. Curran Associates, Inc.

Li, J.; Chen, J.; Ren, R.; Cheng, X.; Zhao, W. X.; Nie, J.-Y.; and Wen, J.-R. 2024. The Dawn After the Dark: An Empirical Study on Factuality Hallucination in Large Language Models. arXiv:2401.03205.

Mi, F.; Wang, Y.; and Li, Y. 2022. Cins: Comprehensive instruction for few-shot learning in task-oriented dialog systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 11076–11084.

Min, B.; Ross, H.; Sulem, E.; Veyseh, A. P. B.; Nguyen, T. H.; Sainz, O.; Agirre, E.; Heintz, I.; and Roth, D. 2023. Recent Advances in Natural Language Processing via Large Pre-trained Language Models: A Survey. *ACM Comput. Surv.*, 56(2).

Mitra, M.; and Chaudhuri, B. 2000. Information Retrieval from Documents: A Survey. *Information Retrieval*, 2(2): 141–163.

Namazifar, M.; Papangelis, A.; Tur, G.; and Hakkani-Tür, D. 2021. Language Model is all You Need: Natural Language Understanding as Question Answering. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 7803–7807.

Peng, B.; Li, X.; Gao, J.; Liu, J.; and Wong, K.-F. 2018. Deep Dyna-Q: Integrating Planning for Task-Completion Dialogue Policy Learning. In Gurevych, I.; and Miyao, Y., eds., *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2182–2192. Melbourne, Australia: Association for Computational Linguistics.

Reimers, N.; and Gurevych, I. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Salton, G. 1983. Introduction to modern information retrieval. *McGraw-Hill*.

Schank, R. C. 1972. Conceptual dependency: A theory of natural language understanding. *Cognitive Psychology*, 3(4): 552–631.

Schatzmann, J.; Weilhammer, K.; Stuttle, M.; and Young, S. 2006. A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies. *Knowl. Eng. Rev.*, 21(2): 97–126.

Scheffler, K.; and Young, S. 2002. Automatic learning of dialogue strategy using dialogue simulation and reinforcement learning. In *Proceedings of HLT*, volume 2, 0.

Schick, T.; and Schütze, H. 2021. It's Not Just Size That Matters: Small Language Models Are Also Few-Shot Learners. In Toutanova, K.; Rumshisky, A.; Zettlemoyer, L.; Hakkani-Tur, D.; Beltagy, I.; Bethard, S.; Cotterell, R.; Chakraborty, T.; and Zhou, Y., eds., *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2339–2352. Online: Association for Computational Linguistics.

Seneff, S. 1992. A relaxation method for understanding spontaneous speech utterances. *Proc. 1992 DARPA Speech and Natural Language Work- shop*, 299–304.

Team, G.; Riviere, M.; Pathak, S.; Sessa, P. G.; Hardin, C.; Bhupatiraju, S.; Hussenot, L.; Mesnard, T.; Shahriari, B.; Ramé, A.; et al. 2024. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.

Väth, D.; Vanderlyn, L.; and Vu, N. T. 2023. Conversational Tree Search: A New Hybrid Dialog Task. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, 1264–1280. Dubrovnik, Croatia: Association for Computational Linguistics.

Väth, D.; Vanderlyn, L.; and Vu, N. T. 2024. Towards a Zero-Data, Controllable, Adaptive Dialog System. In Calzolari, N.; Kan, M.-Y.; Hoste, V.; Lenci, A.; Sakti, S.; and Xue, N., eds., *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, 16433–16449. Torino, Italia: ELRA and ICCL.

Wang, J.; Huang, J. X.; Tu, X.; Wang, J.; Huang, A. J.; Laskar, M. T. R.; and Bhuiyan, A. 2024. Utilizing BERT for Information Retrieval: Survey, Applications, Resources, and Challenges. *ACM Comput. Surv.*, 56(7).

Wei, J.; Bosma, M.; Zhao, V. Y.; Guu, K.; Yu, A. W.; Lester, B.; Du, N.; Dai, A. M.; and Le, Q. V. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

Weld, H.; Huang, X.; Long, S.; Poon, J.; and Han, S. C. 2022. A Survey of Joint Intent Detection and Slot Filling Models in Natural Language Understanding. *ACM Comput. Surv.*, 55(8).

Xu, P.; and Sarikaya, R. 2013. Convolutional neural network based triangular CRF for joint intent detection and slot filling. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, 78–83.

Xu, Z.; Jain, S.; and Kankanhalli, M. 2024. Hallucination Is Inevitable: An Innate Limitation of Large Language Models. arXiv:2401.11817.

Yu, D.; Gong, Y.; Picheny, M. A.; Ramabhadran, B.; Hakkani-Tür, D.; Prasad, R.; Zen, H.; Skoglund, J.; Černocký, J. H.; Burget, L.; and Mohamed, A. 2023. Twenty-Five Years of Evolution in Speech and Language Processing. *IEEE Signal Processing Magazine*, 40(5): 27–39.

Zerveas, G.; Rekabsaz, N.; Cohen, D.; and Eickhoff, C. 2022. CODER: An efficient framework for improving retrieval through COntextual Document Embedding Reranking. In Goldberg, Y.; Kozareva, Z.; and Zhang, Y., eds., *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, 10626–10644. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics.