Combining Discourse Coherence with LLMs for More Robust and Equitable Task-Oriented Dialogue

Katherine Atwell, Mert Inan, Anthony Sicilia, Malihe Alikhani
Northeastern University, Boston, MA
{atwell.ka, inan.m, sicilia.a, m.alikhani}@northeastern.edu

Abstract

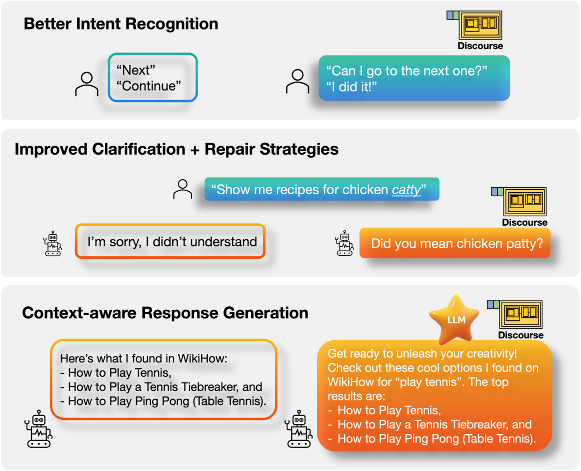
Large language models (LLMs) are capable of generating coherent responses, but using LLMs to synchronously generate responses is not yet feasible for many task-oriented systems, especially those deployed in production environments to achieve user goals. We hypothesize that an offline generation approach using discourse theories, formal grammar rules, and LLMs can allow us to generate human-like, coherent text in a more efficient, robust, and equitable manner within a task-oriented setting. To this end, we present the first discourse-aware multimodal task-oriented dialogue system that combines discourse theories with offline LLM generation. We deploy our system as an app to the general public and keep track of the user ratings for six months. Our user ratings show an improvement from 2.8 to 3.5 out of 5 with the introduction of discourse coherence theories. We also show that our model reduces misunderstandings in the dialect of African-American Vernacular English from 93% to 57%. Further, we find that 100% of the responses generated with our offline generation method were rated *as engaging* or *more engaging* than baseline utterances designed by humans. We release our code in a format that can be integrated into most existing dialogue systems while maintaining compliance with terms of use.

Keywords: task-oriented dialogue, discourse coherence, multimodal , robust

1. Introduction

Effective collaboration between humans and conversational AI requires alignment of goals in each round of dialogue. But it has been shown that large language models (LLMs) can be distracted by irrelevant context (Shi et al., 2023). As powerful as these models are, they struggle in goal-oriented settings. The stochastic nature of LLMs necessitates placing controls on the output in order to provide guarantees on quality, relevance, correctness, or safety. Thus, for deployed task-oriented systems, there is still a need for modular, controllable dialogue systems to generate more effective, human-like, and equitable responses.

In this work, we advocate for an offline approach to dialogue generation grounded in discourse coherence theory (Hobbs, 1979; Hobbs et al., 1985; Kehler, 2006), which provides a theoretical framework for tracking context and goals within a dialogue. It has been argued that discourse coherence theories can be used to interpret sentence fragments within a dialogue (Schlangen and Lascarides, 2003), track speaker commitments (Stone and Lascarides, 2010), and resolve pronouns (Stojnić et al., 2017). We show how, using principles from discourse coherence theories, we simplify the intent recognition problem for our system and develop multiple repair strategies for grounding.


Better Intent Recognition
Discourse
Improved Clarification + Repair Strategies
Discourse
Context-aware Response Generation
LLM
Discourse

"Next"
"Continue"

"Can I go to the next one?"
"I did it!"

"Show me recipes for chicken *catty*"

I'm sorry, I didn't understand

Did you mean chicken patty?

Here's what I found in WikiHow:
- How to Play Tennis,
- How to Play a Tennis Tiebreaker, and
- How to Play Ping Pong (Table Tennis).

Get ready to unleash your creativity! Check out these cool options I found on WikiHow for "play tennis". The top results are:
- How to Play Tennis,
- How to Play a Tennis Tiebreaker, and
- How to Play Ping Pong (Table Tennis).

Figure 1: Three different areas of benefits of using discourse theories for task-oriented dialogue system design. Our model with discourse coherence and LLMs is better at intent recognition, can ask better clarification questions, and can respond with more context awareness. Top box: A model without a discourse framework can only understand simple commands like "next," while our model can understand more complex intents like "I did it." All the utterances are given in pairs, where the filled boxes represent our model's response while the empty boxes represent a simple dialogue system's response without any discourse framework.

Most notably, we use state tracking information from our discourse-aware dialogue management system to fill in response templates generated offline using a formal grammar. This allows us to manually inspect our responses before they are presented to the user, and prevents our system from prompting LLMs with real user data. This is especially important in our domain, in which we

LREC-COLING 2024, pages 3538–3552
20-25 May, 2024. © 2024 ELRA Language Resource Association: CC BY-NC 4.0

intend to use LLMs to take users through intricate, multi-step tasks in real time (i.e., imprecise control of instruction generation can lead to fire hazards, serious illness, or legal/financial trouble).

We present a discourse-aware task-oriented dialogue system, and describe our methods for offline generation using a formal grammar and principles from discourse coherence theories. We highlight our main contributions below:

1. Use contextual features motivated by discourse theories to improve intent recognition and state tracking.
2. Incorporate several critical grounding and repair strategies using state tracking information to achieve common ground with the user.
3. Implement a system of offline LLM generation using a formal grammar and discourse coherence theory for inclusive and human-like generation.

We evaluate our system and find that our discourse-aware approach improves upon our baseline intent model by a large margin, our clarification and repair strategies appear to improve user ratings, and our system mitigates automatic speech recognition (ASR) biases for words or phrases from underrepresented dialects such as African-American Vernacular English (AAVE).

This system was deployed as part of the 2023 Alexa Prize TaskBot Challenge[1] (Agichtein et al., 2023), where thousands of users interacted with our system. We release our code in a manner that is compliant with the terms of the Alexa Prize, while ensuring that our code can be adapted to most existing dialogue systems[2]. The technical paper for our work (Sicilia et al., 2023) is available on the Alexa Prize website.

2. Related Work

Discourse coherence theories describe how units of text relate to the context as a whole, going beyond even the sentence and utterance level. These discourse units are linked to the existing context via *discourse relations*. In some discourse frameworks, these relations are hierarchical (Mann and Thompson; Asher and Lascarides, 2003), while in others, they are "shallow" (only occurring between individual units of text) (Prasad et al., 2008). In this work, we focus on Segmented Discourse Representation Theory (Asher and Lascarides, 2003), a discourse framework in which the dialogue structure is represented as a graph. The first and most well-known SDRT-labeled dialogue corpus, the STAC corpus, is composed of conversations and game moves between players in an online Settlers of Catan game (Asher et al., 2016).

Task-oriented dialogue systems are designed to assist users in completing a particular goal (Fellows et al., 2022). These systems often use external knowledge sources, such as databases and APIs, to aid users in their task (Budzianowski et al., 2018b; Rastogi et al., 2020; Young et al., 2022). These chatbots are typically designed as pipeline systems composed of natural language understanding (NLU), dialogue state tracking (DST), dialogue management (DM), and neural response generation (NRG) modules (Budzianowski et al., 2018b; Rastogi et al., 2020; Zhang et al., 2020). Some systems use an end-to-end approach, but they typically require a large amount of training data, which is expensive to obtain for many task-oriented systems (Zhang et al., 2020), and often require complex neural models. For many tasks and systems in production (including our own), this is infeasible. Thus, in this work we focus on a pipeline approach to response generation using discourse coherence theories.

Task-oriented dialogue systems have become increasingly popular (Budzianowski et al., 2018a; Zhang et al., 2020; Fellows et al., 2022). The closest work to ours is Khalid et al. (2020), who use discourse coherence relations for a referential communication task. Though they find good results, their system is designed for a different setting of collaborative problem-solving and is not a system that assists the user in completing a task, nor is it multimodal. Building on these, we are designing the first system, working with a variety of tasks, combining LLMs and using them for parsing, using discourse for diversity, and proposing creative repair strategies.

Further, we use discourse-level information to deploy repair strategies, often using outputs from LLMs. There are a few works that use discourse relations to control the output of generative models. Balakrishnan et al. (2019) add discourse relations to the input of neural response generation (NLG) models to better control the structure of generated responses. Maskharashvili et al. (2021) find that discourse relations markedly improve the outputs of pretrained NLG models under limited data. However, these works generate responses on the fly, which may be infeasible for systems that cannot host large language models or make external API calls without violating privacy restrictions.

3. Problem Setup

We design a dialogue system with the goal of assisting users in completing various tasks, from

---

[1] https://www.amazon.science/alexa-prize/taskbot-challenge

[2] https://github.com/katherine-atwell/discourse-aware-offline-generation

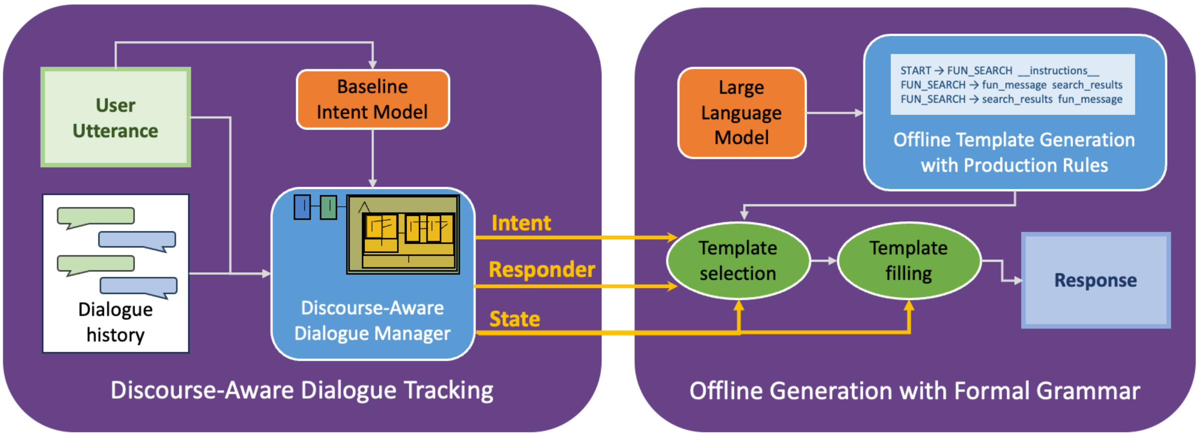

User Utterance
Baseline Intent Model
Dialogue history
Discourse-Aware Dialogue Manager
Intent
Responder
State
Discourse-Aware Dialogue Tracking
Large Language Model
START → FUN_SEARCH __instructions__
FUN_SEARCH → fun_message search_results
FUN_SEARCH → search_results fun_message
Offline Template Generation with Production Rules
Template selection
Template filling
Response
Offline Generation with Formal Grammar

Figure 2: Pipeline for offline natural language generation for our system.

recipes to DIY tasks. To retrieve relevant results, the system queries a Whole Foods database (for recipes) and WikiHow database (for DIY articles) in its backend. These databases were provided to our team by Amazon as part of the Alexa Prize TaskBot Challenge. Once the user has chosen a task, the system provides an overview of the task and, when prompted, reads out each step to the user. Our system is multimodal, and when deployed on a multimodal app-based dialogue system, users can interact with the system via voice input or the touchscreen. We also provide a unimodal dialogue system in the app where users must interact with the system via voice. The user flow is as follows: the users first search for a task, and the bot queries its databases and provides results. The user then chooses a task, and the system provides an overview of the task. Once the user is ready to start, the system reads out each step one by one, allowing the user to complete each step before moving on to the next step. In the following paragraph, we describe how we utilize discourse and context to improve our system.

Progressing through a dialogue with our system requires users to issue a series of *requests*; i.e., in order to view the query results, the user must give or select a search query, and in order to start a task, the user must first select the desired task from the list of query results. At any time, the user may say “stop" to deactivate the bot. The task is considered “complete" when the user indicates that they are done, either by saying “complete" while working on the task or by pressing the *Complete* button that appears at the end of the task. A visualization of the dialogue flow is depicted in Figure 3.

4. Incorporating Discourse into Multimodal Task-Oriented Dialogue

To gain insight on the structure of our system's conversations with users, during the initial stages of system development we manually annotate user conversations using the SDRT discourse framework. An example of one such conversation is in Appendix B. This conversation illustrates a pattern we observe in every successful user dialogue: the dialogue can be split up into four key *stages*, and the interpretation of the user's utterance should depend in part on the current stage. These stages are: *Opening*, *Query & Results*, *Task Info*, and *Completion*, respectively. Figure 3 illustrates these stages in more detail, along with screenshots from the user interface.

We yield additional insights from these manually-annotated discourse graphs. To progress through the above stages, users can only make *subordinating* discourse moves [3] that request a piece of information or specific action from the bot. When the user's request is ambiguous or unsupported at that stage, we wish to *ground* the users, e.g. with *help messages* that let the user know what they can request from the bot. These messages should give examples of things users *can* say to request a particular action from the bot. Thus, when determining what action the bot should take next, we can treat ambiguous intents the same way we treat “invalid" intents (intents that cannot be directly addressed by our bot at that particular point in the dialogue): by grounding the user with a help message or context about a previous utterance. This greatly reduces the number of “valid" intents or responses the model must check for at each turn.

By tracking the dialogue context and focusing on desired discourse *relations* between our bot's utter-

---

[3] ones which do not progress the conversation further

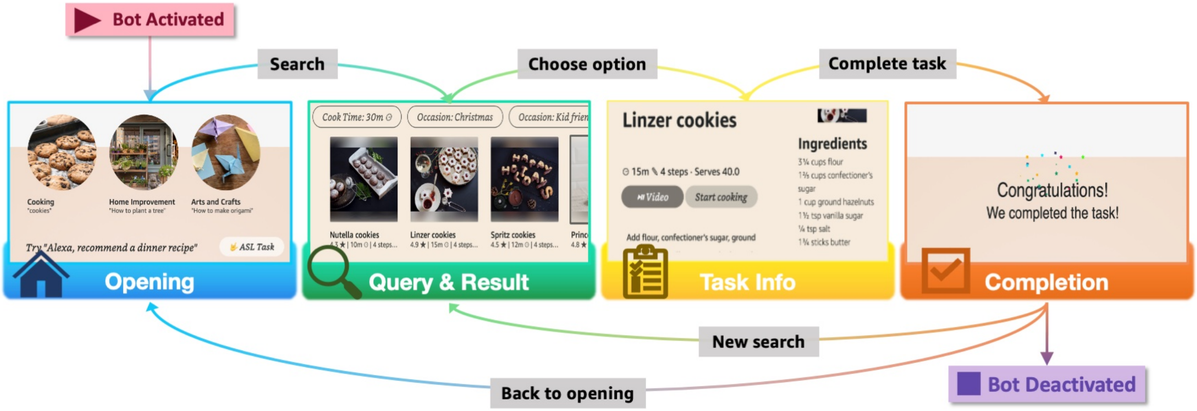


Figure 3: Diagram depicting the user flow for our TaskBot, with screenshots of the UI at each stage. The actions depicted in the arrows represent the user actions that must take place in order to progress to the next stage.

User Utterance	Stage	Baseline Intent Label	Context-Aware Intent Label
i'm thinking of blueberry muffins	Opening	`General`	`QueryIntent`
option two	Query & Results	`General`	`SelectIntent`
the last step	Task Info	`General`	`PreviousIntent`

Table 1: Instances where the baseline intent model does not predict the user intent correctly, but the contextually-aware intent model does (note: these were not taken from real user conversations)

ance and the previous user utterance, we can thus constrain the number of candidate intents at each turn and simplify the question of 1) what the user's intent is (§4.1) and 2) whether to deploy a grounding strategy (§4.1.1). We next introduce a novel formal grammar that allows us to generate response candidates offline and incorporate contextual information during runtime (§4.2). This reduces repetition and improves human-likeness in our bot's responses, while at the same time minimizing human effort and allowing for complete control over the generated text. We illustrate this approach in Figure 2, and describe our methods in detail in the following sections. Note that, to preserve user privacy, none of the sample dialogues or utterances in this paper are pulled from real user dialogues. Rather, they are based on common user queries and requests, and multi-turn dialogues are simulated using our bot.

4.1. Context-Aware Intent Recognition

When the user's utterance is first processed, a baseline few-shot intent recognition and slot-filling model is run, which labels intents (and slots, if applicable) based on the utterance's similarity to a set of samples. Some intents can be easily recognized by the baseline model, regardless of context, such as Yes, No, and StopIntents. After running the baseline model, we use dialogue state information to run our context-aware intent recognition model. Depending on the baseline intent and the dialogue context, this may overwrite the baseline intent model. We provide some examples of this in Table 1. In the following, we describe several heuristics used by our context-aware model to classify intents.

Opening When the user is in the Opening stage, the only valid moves the user can make are to provide their own search query or select a search query from the menu. Thus, if the system determines that the user is not selecting an item (e.g. "the first one"), it uses a series of heuristics (which were iteratively designed based on user conversations, domain knowledge, and article titles) to determine whether the user's utterance is likely to be a query. For instance, if the user utters a noun phrase (e.g. "cake recipes"), or uses a verb that commonly occurs in how-to article titles (e.g. "pick a lock"), the bot labels their utterance as a search query and performs slot-filling using similar heuristics.

Task Info During the Task Info stage, as with each dialogue state, there is a set of intents deemed "acceptable" for the user - intents that yield discourse relations with salient utterances. For instance, if a bot has just finished reading out a step to the user, this step is considered *salient*, as is the task

User Query	Response
thanksgiving recipes	These recipes are sure to make your taste buds happy! Check out what I found on the Whole Foods market place for your search on thanksgiving recipes: The top results are Squash and cranberry stuffing with sage butter, Chocolate pecan pie, and Granny's sweet potatoes with peanut butter and marshmallows. If you want the highest rated, check out the first one, and for a quick recipe, check out the third choice. Don't forget to select an option, summarize it, or search for something else!
how to play tennis	Get ready to unleash your creativity! Check out these cool options I found on WikiHow for play tennis: The top results are How to Play Tennis, How to Play a Tennis Tiebreaker, and How to Play Ping Pong (Table Tennis). You can select an option, summarize an option, or search something else.
start a garden	Let's make something awesome! Check out these ideas I found on WikiHow for start a garden: The top results are How to Start a Flower Garden, How to Start a Vegetable Garden, and How to Start a Garden. You can select an option, summarize an option, or search something else.

Table 2: Examples of bot responses generated from a formal grammar to list search results, as shown in §4.2 (note: these were not taken from real user conversations).

overview. The requests a user is allowed to make at this stage must relate to one of the salient bot acts: either the current step or the overview of the task as a whole. These include *going to the next step*, *going back to the previous step*, *going back to the task summary page*, and *completing the task*. We use a simple rule-based classifier that uses keyword matching strategies to predict these intents when the baseline model fails, for instance "I did it" or "I want to go to the next step". By constraining the user intent candidates to those that form coherent relations with salient dialogue context, we are able to predict user intents with higher accuracy (§ 5).

4.1.1. Choosing Repair Strategies

Below, we detail common repair strategies deployed in our system (see Appendix C for examples).

Clarification questions Clarification questions clarify a user's goals, often in cases of suspected automatic speech recognition (ASR) errors. If a user's intent or selection cannot be determined, the bot compares the user's utterance against a set of contextually-relevant candidates. If one of the candidates is similar to the user's utterance, the bot will ask the user if they meant that.

Question tracking When the bot has posed a question that has not been answered by the user, the user must provide a satisfactory answer to the question for the dialogue to progress. If a satisfactory answer is not provided, or the user's intent cannot be ascertained, the bot repeats the question and outlines the types of responses that are valid (e.g. "yes"/"no" answers). We refer to this grounding strategy as *question context*.

Help messages The user intent cannot always be mapped to one of the valid intent candidates, whether because of an intent recognition error or an unsupported user action at that stage. To handle this, the bot generates *help messages*, which tell the user that their utterance appeared not to fall within the allowed intents, and suggest things the user can say to facilitate grounding. These messages are generated using our offline generation method, which we describe in the next section.

4.2. Context-Aware LLM-based Natural Language Generation

Using our discourse-aware system and the dialogue management strategies above, we are able to provide relevant responses for the user based on the existing dialogue context. But how can we make these responses human-like and prevent them from sounding repetitive? One solution is to use LLMs "on the fly", automatically generating live responses to a user's utterance. Recent utterance-level performance improvements and human-likeness of end-to-end LLMs such as GPT (Brown et al., 2020), PaLM (Chowdhery et al., 2022), LLaMa (Touvron et al., 2023), and AlexaTM (Soltan et al., 2022). However, making API calls to a closed-source LLM to generate a relevant response requires exposing user utterances, and details of the conversation, to a third-party system. Open-source alternatives, meanwhile, are costly to house or to train. Regardless of the method, responses generated on the fly cannot be manually inspected for quality, relevance, and safety before they are presented to the user. Thus, the responses could contain errors or harmful content, or may not be relevant to the conversation or user goals.

One way to mitigate this problem is to generate LLM utterances offline, manually validating candidates before putting them into production. For many dialogue acts, this strategy can be as simple

Stage	Message
Opening	I'm sensing you might be a bit lost, or I could not understand you. This is our landing page. What do you want to do together today? Try saying I want to start a garden.
Query & Results	It looks like we need some help getting started. You can tap on the options on the screen. You'll need to select one of the tasks I have found to start working together. So, which task do you want to do? As a reminder: Here are your previous options. The top results are How to Solve a Cryptic Crossword, How to Finish a Crossword Puzzle, and How to Make Crossword Puzzles. You can select an option, summarize an option, or search for something else.
Task Info	It seems like either you or I might need a little guidance on what to do next. I was talking about the steps of our task. If you want to hear it again, you can ask me to repeat it.

Table 3: Examples of help messages generated from a formal grammar for different stages, as described in §4.2

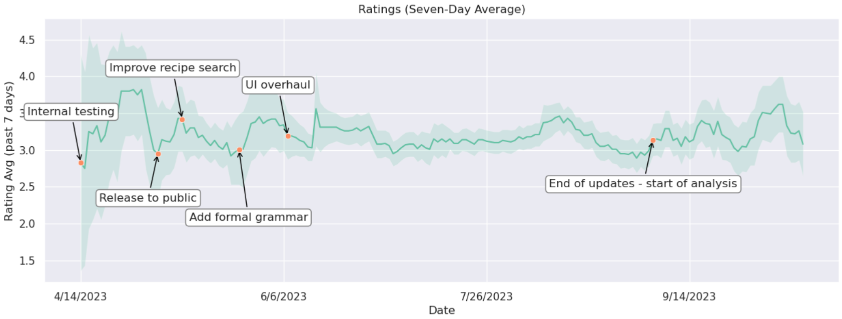


Figure 4: Seven day averages for user ratings over the course of deployment. Notable events, such as changes to the user base or features added, are marked on the dates they occurred. On September 5th, we stop adding new features, and we analyze conversations from this day onward.

as generating preambles, postambles, and other stand-alone utterances, then inserting them programmatically into the symbolically planned text outputs (i.e., planned using our discourse-aware intent and management strategies). Unfortunately, this limits the structure and diversity of the generated text, making it infeasible for dialogue acts which are more context-sensitive. The main challenge arises because of key lexical, semantic, and structural constraints imposed on the generation by the system's symbolic components:

1. A symbolic algorithm will process the users' next utterance, so clear communication of possible "next steps" to the user and how to trigger these steps is important.
2. Programmatic insertion of outputs from the system's symbolic generation components must be possible, e.g., retrieved articles and attribution in the case of displaying search results.
3. Any necessary components of the generation (e.g., content attribution) should be present.
4. The user's query cannot be provided to the (third-party) LLM to preserve user privacy.

To accommodate generation under these constraints, we propose to use *formal grammars* (Chomsky, 1956). A formal grammar is a model of language defined by the following components.

Terminals: a set which defines the base alphabet or vocabulary for the grammar (we modify this slightly to increase stochasticity, asking the LLM to generate these itself)

Variables: can take on different combinations from the terminal alphabet

Production Rules: describe how to go from the **start** symbol (a special variable) to other variables, and then to the terminals, which in the end, should form a full utterance in the grammar.

We propose two strategies using formal grammars to generate offline: (1) we directly prompt `gpt-3.5-turbo` with the grammar's rules, asking this LLM to generate according to these rules or (2) we use `gpt-3.5-turbo` to generate *terminals* only, sampling from the grammar based on the production rules and variables programmatically. These approaches are discussed in greater detail, and prompt templates are released in full, in Appendix D.1 and 4.2.1, respectively. Examples of outputs produced by our method for search results and help messages can be found in Table 2 and 3.

4.2.1. Using GPT-3.5 to Generate Terminals Only

Help messages guide the users when they are lost, and thus require contextual awareness to let the user know what actions they can perform based on the current state of the conversation. To generate these messages, we employ a set of grammatical rules and templates. They are based on the current state and the response that is shown to the user.

Similar to the above section, we generate help messages based on formal grammatical rules. For help messages, rather than supplying rules *within* the prompt to an LLM like `gpt-3.5-turbo`, we instead use human effort to design the grammar's production rules and variables, automating only the generation of terminals (i.e., using `gpt-3.5-turbo` with some human supervision). The randomly generated terminals can then be combined using simple deterministic algorithms.

For example, one grammatical rule for generating help messages is the following:

> `start` → `confused` + `eurekaIntj` + `info` + `whatWantDoQ`

where '+' denotes concatenation and all other terms are variables. `gpt-3.5-turbo` can then generate multiple terminals that map directly to each variable with human supervision; e.g.,

> `confused` → Hmm, it looks like we need some help getting started.
> `confused` → You don't know what to do, right?
> `confused` → I'm having trouble understanding you.
> ...

After randomly generating terminals, we can execute rules with a simple deterministic algorithm. So, a full help message corresponding to the rule from before could look like:

> Hmm, it looks like we need some help getting started. Aha! You can ask me to help you make a recipe or do a DIY task. What would you like to do today?

A high degree of stochasticity comes from having `gpt-3.5-turbo` generate multiple potential terminal values for each variable, making the help messages appear more human-like.

5. Results

We comprehensively evaluate our system by deploying it to users over six months and collecting overall satisfaction ratings. Our system was deployed to thousands of users over the course of this six-month period as part of the Alexa Prize TaskBot Challenge, where it was advertised to users with Echo Show devices. We use these ratings and user conversations to iteratively improve our system's capabilities to adapt to users' needs during the first five months and evaluate our system over the last month of deployment. We find evidence suggesting that our clarification strategies improve user experience and that incorporating conversational context into our system improves our intent model. Our evaluations also indicate that our neurosymbolic, offline generation approach yields varied, human-like utterances without needing external API calls.

Below, we show the progression of the user ratings throughout the deployment, highlighting dates where significant feature changes and user demographic changes occurred. We then provide a regression analysis of user data from the last month of deployment (where no additional changes to the bot were made) to evaluate the effectiveness of our discourse- and context-aware grounding strategies. Finally, we use manual annotations to analyze the effectiveness of our strategy for intent modeling and language generation. We present these results below, and provide further discussion in §6.

5.1. User Ratings

We track the user ratings throughout the deployment of our system, starting April 14, 2023, and ending October 12, 2023. The users are asked to rate their overall satisfaction after interacting with the bot on a scale from 1 to 5 and provide additional feedback if applicable.

Figure 4 showcases how these scores evolved throughout the deployment. It is marked with relevant events, including the beginning of internal testing and our bot's release to the general public.

We observe that ratings improve after 1) releasing the bot to a general audience, 2) incorporating our generation strategy using formal grammar, and 3) the UI overhaul. Note that changes may take up to a week to be reflected on the graph, which plots seven-day averages. Further, the makeup of the system's user base changed throughout the deployment, as Amazon ran advertising campaigns throughout which included themed events. An example of this is the improvement in results during internal testing, followed by a significant drop in ratings when the system was released to the public. Because the system was advertised to a large number of users with Echo Show devices, users were frequently directed to the Alexa Prize TaskBot Challenge bots without a clear idea of their capabilities. An example of this which was commonly found in conversational transcripts occurred when users asked Alexa to turn on the lights, check the time, or play a video. Due to the presence of these confounders throughout the competition, in §6 we perform a regression analysis on the last month of ratings (starting with "End of Updates" on the ratings plot) to analyze the impact of individual features more closely.

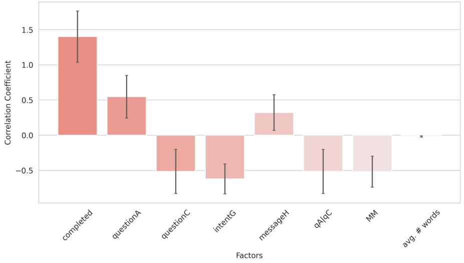


Figure 5: Linear regression model predicting user ratings based on the above factors.

5.2. Regression Analysis

We conduct a regression analysis to disentangle the effects of different features on user ratings. For this analysis, we use the `statsmodels` package. We analyze the set of rated conversations between September 4 and October 4 for a total of 569 rated conversations. Each data point represents a conversation, and our coefficients are calculated at the conversation level. Although we experimented with both linear and non-linear models, we showcase the results of our linear regression analysis, which resulted in the greatest reduction of error. We present the results of our linear regression analysis in Figure 5. We briefly describe each coefficient below.

completed: whether the user completed a task during the conversation.

question asked: whether the bot asked a question at some point during the conversation

question context: whether the bot had to repeat a question to a user at any point during the conversation (see the last bot response in the Question Tracking Figure 7).

general intent predicted by baseline: whether the baseline (non-contextually-aware) intent model predicted a “general" intent during the conversation

help message: whether the bot issued a help message during the conversation (Figure 7)

multimodal: whether the user was using a multimodal device (as opposed to headless)

average number of words: average number of words the user utters per turn during the conversation

We are interested in measuring the efficacy of our grounding strategies. **completed**, **general intent predicted by baseline**, **multimodal**, and **average number of words** serve as potential confounders which we control for. These results show that the coefficients most likely to improve ratings are **completed** and **question asked**. We discuss these results, and their implications, more in §6.

We also find evidence that help messages may improve ratings, but these results are insignificant.

5.3. Manual Evaluation

We perform manual annotations for the intent recognition and generation modules to measure the specific performances of our context-aware models.

Context-Aware Intent Labeling To evaluate our context-aware intent labeling strategies described in §4.1 compared to the baseline intent model, utterances from user conversations are randomly selected and stratified between positive and negative predictions so an equal amount of each is labeled. Our contextual model shows an improved average between precision and recall for each intent, as well as improvements in precision and recall for each intent except QA (precision) and Select (precision) (Table 4).

LLM Generation with Formal Grammar Using the formal grammar rules described in §4.2, we prompt `gpt-3.5-turbo` to output 70 responses designed to convey search results to a user. Our manual evaluation of LLM-generated text using formal grammar shows that 86% of generated utterances met all of the constraints given in our grammar. We also found that 100% were *as engaging* or *more engaging* than baseline utterances designed by humans. Further, themes or partial utterances were repeated across only 31% of the generations based on human evaluation.

6. Discussion

Here, we interpret our results from the previous section and enumerate the benefits of our discourse-aware strategies for intent recognition, user satisfaction, and human-likeness. We also discuss the impacts on inclusivity, privacy, and safety.

Contextually-aware intent labeling improves intent recognition and may mitigate bias Based on the results in Table 4, our contextually-aware intent model outperforms the baseline by a large margin for both precision and recall. Thus, utilizing our context-aware model improves intent recognition performance over using a baseline context-agnostic model. In the following, we examine how a speaker’s dialogue may affect these results.

It is known that current automatic speech recognition (ASR) systems hold a number of racial, cultural, and social biases, as well as biases towards women and disabled people (Ngueajio and Washington, 2022). We perform a preliminary manual evaluation to determine the accuracy of the intent model for the African-American Vernacular English (AAVE) keyword “wanna", compared to the standard English counterpart, “want to".

We find that the baseline intent model correctly recognizes the intent for the word “want to" 100%

		Query	Select	QA	Back	Help	Step Nav.	Stop	Complete
Without Context	**P**	0.66	1.0	0.50	0.56	0.00	0.83	0.33	0.67
	R	0.48	0.21	0.06	0.56	0.00	0.47	0.41	0.51
With Context	**P**	0.70	0.74	0.30	0.74	0.42	0.91	0.77	0.74
	R	0.88	0.97	0.88	0.60	0.91	0.67	0.45	0.72

Table 4: Precision (**P)** and Recall (**R**) for user intent recognition (**W/**) and without (**W/O**) our discourse-aware approach, based on human evaluation of a random sample of user utterances.

of the time, but only 7% for “wanna" - a 93% performance gap. Our context-aware intent model, however, classifies the correct intent for 43% of the “wanna" instances, while maintaining 100% correctness for the “want to" instances. This drops the performance gap from 93% to 57%, a 36% reduction. As this evaluation shows, there still exists a significant performance gap between these two keywords; however, deploying our model reduces this gap.

Clarification strategies improve user satisfaction Our regression analysis results (Figure 5) indicate that asking a question to the user significantly improves ratings ($p > 0.1$) by over 0.5 points. We disambiguate the instances where question context must be provided, as this indicates that the user answered the bot’s question incorrectly and may be uncooperative or become frustrated. We find that providing question context leads to a significant decrease in ratings when conditioned upon question-asking, but question-asking as a whole improves ratings.

Discourse allows us to incorporate LLMs to generate human-like dialogue in a privacy-preserving manner In Figure 4, we see that adding formal grammar to our model results in an upswing in user ratings. As this phenomenon could be due to multiple factors, we perform additional human evaluation to determine the human-likeness of the generated text. From the results in Section 5.3, we find that the generated text is able to conform to the constraints of the grammar, while still being human-like and avoiding repetition.

We also view the regression results in Section 5.2 to be a positive sign, given how high the **help messages** coefficient is. While this effect size is not statistically significant, the ordinary least squares coefficient estimates we use are unbiased.[4]

By prompting large language models to create templates offline without user data, and using conversational context to fill in the templated responses in real-time, we make the best use of LLMs without exposing user information to third-party APIs. This allows systems that cannot host language models themselves to preserve user privacy while still producing varied, human-like responses.

Our discourse-aware strategies result in a safe, reliable system Along with quality control issues, LLM-generated text comes with safety concerns (Zhiheng et al., 2023). Because we are not generating task information on the fly, our main vulnerabilities regarding safety are 1) how we respond to harmful or dangerous user queries and 2) whether any safety issues exist in the articles we show to users. At any point during our system’s deployment, if dangerous content is shown to a user, our bot is deactivated for 24 hours and can be reactivated only after a fix is made that prevents this content from showing up in the future. To address potential safety issues, we use different techniques for different contexts; we use classifiers to detect dangerous user queries, and we have separate lists of guardrails for article titles and article steps. These guardrails were developed based on detailed searches of the databases, and were iteratively added as needed over the course of deployment. Using our safety techniques in combination with the rest of our techniques, we were able to achieve 93% uptime over the course of deployment.

7. Conclusion

In this work, our goal was to study how discourse coherence theories in combination with LLMs can empower task-oriented dialogue systems to be more robust, aligned with user goals, and effective with diverse populations. Using a combination of manual analysis and user ratings collected over six months, we find support for our approach, which yields improvements in intent recognition and user ratings while reducing manual labor required to create response candidates. Including more theory-informed approaches in multimodal interactive systems in the future will enable us to explore the potential of LLMs in designing human-level conversational systems.

---

[4] This is true whenever the mean of the error distribution is 0, which we verify empirically: the mean error of our model is xyz.

8. Acknowledgements

We would like to thank the Alexa Prize Taskbot Challenge team, especially Lavina Vaz, and Michael Johnston for helping us throughout the competition, and for giving us the opportunity and the resources to develop and deploy our system to a large audience of users. We would also like to thank our team members: Yuya Asano, Sabit Hassan, Dipunj Gupta, Paras Sharma, Jennifer Nwogu, and Qi Cheng.

9. Bibliographical References

Eugene Agichtein, Michael Johnston, Anna Gottardi, Lavina Vaz, Cris Flagg, Yao Lu, Shaohua Liu, Sattvik Sahai, Giuseppe Castellucci, Jason Ingyu Choi, Prasoon Goyal, Di Jin, Saar Kuzi, Nikhita Vedula, Lucy Hu, Samyuth Sagi, Luke Dai, Hangjie Shi, Zhejia Yang, Desheng Zhang, Chao Zhang, Daniel Pressel, Heather Rocker, Leslie Ball, Osman Ipek, Kate Bland, James Jeun, Oleg Rokhlenko, Akshaya Iyengar, Arindam Mandal, Yoelle Maarek, and Reza Ghanadan. 2023. Advancing conversational task assistance: the second alexa prize taskbot challenge. In *Alexa Prize TaskBot Challenge 2 Proceedings*.

Nicholas Asher and Alex Lascarides. 2003. *Logics of conversation*. Cambridge University Press.

Anusha Balakrishnan, Jinfeng Rao, Kartikeya Upasani, Michael White, and Rajen Subba. 2019. Constrained decoding for neural NLG from compositional representations in task-oriented dialogue. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 831–844, Florence, Italy. Association for Computational Linguistics.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 2018a. 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.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gasic. 2018b. 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.

Noam Chomsky. 1956. Three models for the description of language. *IRE Transactions on information theory*, 2(3):113–124.

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.

Ryan Fellows, Hisham Ihshaish, Steve Battle, Ciaran Haines, Peter Mayhew, J Ignacio Deza, et al. 2022. Task-oriented dialogue systems: Performance vs quality-optima, a review. In *CS & IT Conference Proceedings*, volume 12. CS & IT Conference Proceedings.

Jerry R Hobbs. 1979. Coherence and coreference. *Cognitive science*, 3(1):67–90.

Jerry R Hobbs et al. 1985. *On the coherence and structure of discourse*, volume 208. CSLI Stanford, CA.

Julie Hunter, Nicholas Asher, and Alex Lascarides. 2018. A formal semantics for situated conversation. *Semantics and Pragmatics*, 11:10–EA.

Andrew Kehler. 2006. Discourse coherence. *The handbook of pragmatics*, pages 241–265.

Baber Khalid, Malihe Alikhani, Michael Fellner, Brian McMahan, and Matthew Stone. 2020. Discourse coherence, reference grounding and goal oriented dialogue. In *Proceedings of the 24th Workshop on the Semantics and Pragmatics of Dialogue*.

William C Mann and Sandra A Thompson. Rhetorical structure theory: Description and construction of text structures. In *Natural language generation: New results in artificial intelligence, psychology and linguistics*, pages 85–95. Springer.

Aleksandre Maskharashvili, Symon Stevens-Guille, Xintong Li, and Michael White. 2021. Neural methodius revisited: Do discourse relations help with pre-trained models too? In *Proceedings of the 14th International Conference on Natural Language Generation*, pages 12–23, Aberdeen, Scotland, UK. Association for Computational Linguistics.

Mikel K Ngueajio and Gloria Washington. 2022. Hey asr system! why aren't you more inclusive? automatic speech recognition systems' bias and proposed bias mitigation techniques. a literature review. In *International Conference on Human-Computer Interaction*, pages 421–440. Springer.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind Joshi, and Bonnie Webber. 2008. The Penn Discourse TreeBank 2.0. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco. European Language Resources Association (ELRA).

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. 2020. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8689–8696.

David Schlangen and Alex Lascarides. 2003. The interpretation of non-sentential utterances in dialogue. In *Proceedings of the Fourth SIGdial Workshop of Discourse and Dialogue*, pages 62–71.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H Chi, Nathanael Schärli, and Denny Zhou. 2023. Large language models can be easily distracted by irrelevant context. In *International Conference on Machine Learning*, pages 31210–31227. PMLR.

Anthony Sicilia, Yuya Asano, Katherine Atwell, Qi Cheng, Dipunj Gupta, Sabit Hassan, Mert Inan, Jennifer Nwogu, Paras Sharma, and Malihe Alikhani. 2023. Isabel: An inclusive and collaborative task-oriented dialogue system. In *Alexa Prize TaskBot Challenge 2 Proceedings*.

Saleh Soltan, Shankar Ananthakrishnan, Jack FitzGerald, Rahul Gupta, Wael Hamza, Haidar Khan, Charith Peris, Stephen Rawls, Andy Rosenbaum, Anna Rumshisky, Chandana Satya Prakash, Mukund Sridhar, Fabian Triefenbach, Apurv Verma, Gokhan Tur, and Prem Natarajan. 2022. Alexatm 20b: Few-shot learning using a large-scale multilingual seq2seq model.

Una Stojnić, Matthew Stone, and Ernie Lepore. 2017. Discourse and logical form: Pronouns, attention and coherence. *Linguistics and Philosophy*, 40:519–547.

Matthew Stone and Alex Lascarides. 2010. Coherence and rationality in grounding. In *Proceedings of the 14th Workshop on the Semantics and Pragmatics of Dialogue*, pages 51–58. Poznán Poland.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models.

Tom Young, Frank Xing, Vlad Pandelea, Jinjie Ni, and Erik Cambria. 2022. Fusing task-oriented and open-domain dialogues in conversational agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 11622–11629.

Zheng Zhang, Ryuichi Takanobu, Qi Zhu, MinLie Huang, and XiaoYan Zhu. 2020. Recent advances and challenges in task-oriented dialog systems. *Science China Technological Sciences*, 63(10):2011–2027.

Xi Zhiheng, Zheng Rui, and Gui Tao. 2023. Safety and ethical concerns of large language models. In *Proceedings of the 22nd Chinese National Conference on Computational Linguistics (Volume 4: Tutorial Abstracts)*, pages 9–16, Harbin, China. Chinese Information Processing Society of China.

10. Language Resource References

Nicholas Asher, Julie Hunter, Mathieu Morey, Benamara Farah, and Stergos Afantenos. 2016. Discourse structure and dialogue acts in multiparty dialogue: the STAC corpus. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 2721–2727, Portorož, Slovenia. European Language Resources Association (ELRA).

A. The STAC Corpus

The STAC corpus is an SDRT-labeled corpus which contains conversations and game moves from online Settlers of Catan games (Asher et al., 2016). Thus, the STAC corpus contains relations between linguistic (chats) and non-linguistic acts (game moves). Table 5“‘ shows an example from Hunter et al. (2018) of such relations; in turn 434, for instance, GWFS types “noooo!", referring to the game move depicted in 433.0.4, where william monopolizes wheat. As such, a *Comment* relation holds between the game move (a nonlinguistic event) and the utterance (a linguistic event). In the STAC corpus, all game moves are part of the “core" of the discourse structure, and would all remain connected to one another even if the chat moves were removed from the graph. However, as we will explain further below, some linguistic and nonlinguistic events from the user may represent the same moves when interacting with our TaskBot. Further, many TaskBot turns contain a combination of linguistic utterances and multimodal displays within the same turn.

433.0.3	Server	william played a Monopoly card
433.0.4	Server	william monopolized wheat
433.0.5	Server	It's william's turn to roll the dice.
434	GWFS	noooo!
435	Server	william rolled a 2 and a 1.
436	Server	GWFS gets 1 sheep. LJAY gets 2 wood. TK gets 2 wood.
436.0.0.1	UI	GWFS has 4 resources. LJAY has 3 resources. william has 13 resources.
437	GWFS	greedy :D
438	william	:D
439	GWFS	spend it wisely then
440	LJAY	:)
441	LJAY	13! :o

Table 5: A sample conversation from the STAC corpus (Asher et al., 2016) in which discourse relations hold between linguistic events (e.g. chats) and non-linguistic events (e.g. game moves)

B. A Discourse Analysis of Our System's Task-Oriented Dialogues

In annotating the discourse structure of our system's dialogues, we arrive at several conclusions. We describe these conclusions below, and delve into a conversation that exemplifies these conclusions.

1. There can only exist *subordinating* relations (relations which do not move the dialogue forward) between a user's utterance and the preceding dialogue. This is consistent with the nature of our task, as only the TaskBot's utterances move the dialogue forward and the user's utterances serve as questions or requests for information.

2. Each TaskBot stage connects to the previous stage via a *Result* relation.

3. Turns which involve changes to the multimodal interface must hold a *coordinating* relation (relations which move the dialogue forward) with a discourse unit in the preceding dialogue.

A sample conversation between a user and our bot can be found in Table 6, and the corresponding discourse graph (using the SDRT framework) can be found in Figure 6.

C. Grounding Strategy Examples

D. LLM-generated Responses

D.1. Directly Prompting GPT-3.5

Below is an example for how we can directly prompt `gpt-3.5-turbo` using a formal grammar. Interestingly, GPT tends to understand the grammar as well as our slight modifications.

Example Prompt: *(temp: 1, top p: 0.9, freq penalty: 0.25)*

```
  I am going to describe a formal
grammar and then ask you to gener-
ate text according to this grammar.
To promote more diverse derivations
from the grammar, we will modify the
usual definition so that terminals
can be stochastic.  To achieve this
behavior, a similarity symbol "∼"
will mean you should self-prompt,
so you are supposed to generate the
terminal value on your own according
to the provided prompt.  For more
traditional, deterministic terminal
assignments in the grammar, we will
use the equality "=" symbol.  These
terminals should be copied exactly.
```

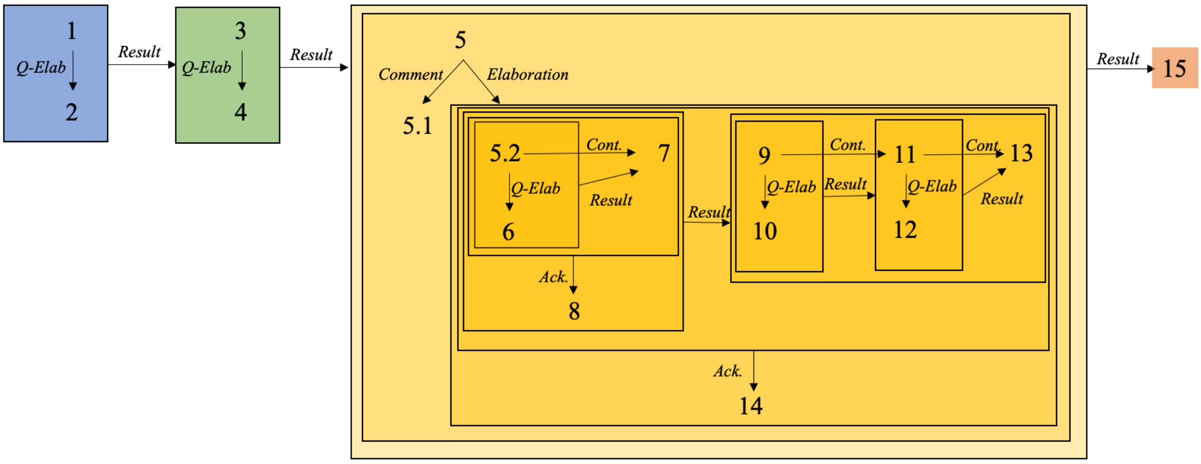
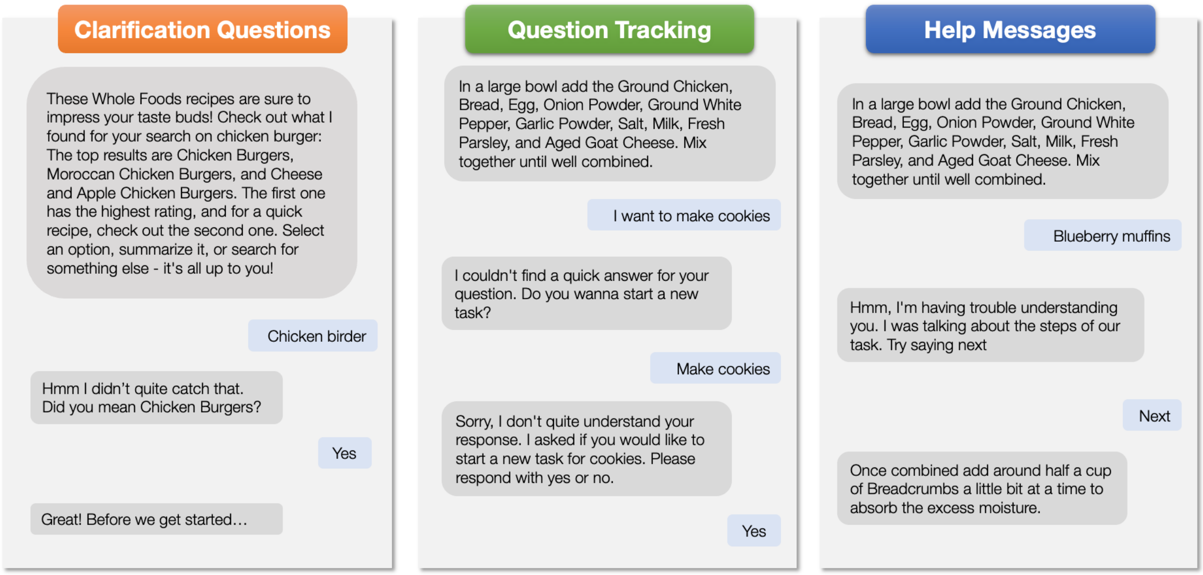


Figure 6: SDRT graph for the sample interaction between our TaskBot and a user detailed in Table 6. The four colors correspond to the four stages of our TaskBot, as depicted in Figure 3.



These Whole Foods recipes are sure to impress your taste buds! Check out what I found for your search on chicken burger: The top results are Chicken Burgers, Moroccan Chicken Burgers, and Cheese and Apple Chicken Burgers. The first one has the highest rating, and for a quick recipe, check out the second one. Select an option, summarize it, or search for something else - it's all up to you!

Chicken birder

Hmm I didn't quite catch that.
Did you mean Chicken Burgers?

Yes

Great! Before we get started…

In a large bowl add the Ground Chicken, Bread, Egg, Onion Powder, Ground White Pepper, Garlic Powder, Salt, Milk, Fresh Parsley, and Aged Goat Cheese. Mix together until well combined.

I want to make cookies

I couldn't find a quick answer for your question. Do you wanna start a new task?

Make cookies

Sorry, I don't quite understand your response. I asked if you would like to start a new task for cookies. Please respond with yes or no.

Yes

In a large bowl add the Ground Chicken, Bread, Egg, Onion Powder, Ground White Pepper, Garlic Powder, Salt, Milk, Fresh Parsley, and Aged Goat Cheese. Mix together until well combined.

Blueberry muffins

Hmm, I'm having trouble understanding you. I was talking about the steps of our task. Try saying next

Next

Once combined add around half a cup of Breadcrumbs a little bit at a time to absorb the excess moisture.

Figure 7: Examples of conversations between a user and our bot that exemplify each of the grounding strategies mentioned in Section 4.1.1: clarification questions, question tracking, and help messages (note: these were not taken from real user conversations).

The set of stochastic terminals is below. Remember, you should generate these according to the self-prompt:

- fun_message ∼ generate something exciting to hype a user up before they start working on a DIY task. For example, "Are you ready to get started" or "These options seem fun!" or "This looks like it will be a blast!"
- search_results ∼ generate a message to display WikiHow article options that were found on WikiHow. Make sure to attribute WikiHow and use the placeholder "___input_search___" and "__list_options__" to indicate what was searched and what was found, respectively. For example, "Here's what I found on the WikiHow for ___input_search___: __list_options__." Copy both placeholders verbatim.

The set of deterministic terminals

1*	Bot	Welcome! I can help with a task you choose. You can ask me for things like recipes for cookies, how to garden, or how to make origami.
2	User	I want to bake a cake
3*	Bot	Are you excited to try some new recipes? Whole Foods has got you covered! For your search on a cake, I found some great options: The top results are Carrot Cake Coffee Cake, Louisa's Cake, and Hummingbird Cake. . Now, it's up to you to select an option, summarize it, or search for something else.
4	User	the first option
5*	Bot	*Displays summary view of recipe*
5.1	Bot	Great! Before we get started, please be careful when using any tools or equipment. Remember, safety first!
5.2	Bot	OK, here are the ingredients for Carrot Cake Coffee Cake. The first ingredient is: 2 large carrots. Say 'next' when you're ready for another. Or say 'start cooking' when you're ready for the recipe steps.
6	User	next
7	Bot	¾ cup granulated sugar
8	User	start cooking
9*	Bot	Preheat the oven to 350 degrees. Peel and grate the carrots (I use my food processor). You should have about 1 1/2 cups.
10	User	the next step
11*	Bot	Whisk sugar, melted butter, and eggs. Stir in carrots. Add flour, baking soda, and cinnamon. Mix until just combined. The batter will be very thick.
12	User	next
13*	Bot	Mix ingredients for the streusel until you get a texture that looks like pebbles (sometimes I use my hands to mix it all together).
14	User	complete
15*	Bot	Awesome! Task complete! If you want to start another task, just ask me.

Table 6: Sample interaction between our bot and a user, where the user selects a recipe and the bot guides the user through it step-by-step (note: this is not a real user conversation). The discourse graph of this interaction can be found in Figure 6. A * indicates that a change to the bot's multimodal interface occurs as part of that turn.

is below. Recall, these should be copied exactly as written here:

- instructions = "You can select an option summarize an option or search something else."

The set of variables are:

- FUN_SEARCH
- INSTRUCT

The start symbol is:

- START

The set of production rules are:

- START -> FUN_SEARCH __instructions__
- FUN_SEARCH -> fun_message search_results
- FUN_SEARCH -> search_results fun_message

An example derivation is given below:

Mmm, these recipes look delicious! Here's what I found on the Whole Foods marketplace for __search__: __list_options__. You can select an option summarize an option or search for something else.

Generate 10 diverse and fun examples according to this grammar. Respond differently to the self-prompts each time. Use all the rules with equal likelihood.

Example Output:

- Are you ready for some DIY fun? Here's what I found on WikiHow for __input_search__: __list_options__. You can select an option, summarize an option, or search for something else.
- Oh boy, this is going to be exciting! I found some awesome ideas on WikiHow for __input_search__: __list_options__. You can select an option, summarize an option, or search for something else.
- These options seem like a blast! Check out what I found on WikiHow for __input_search__: __list_options__. You can select an option, summarize an option, or search for something else.

- Get ready to unleash your creativity! Check out these ideas I found on WikiHow for __input_search__: __list_options__. You can select an option, summarize an option, or search for something else.
- Let's see what we've got! Here are some options I found on WikiHow for __input_search__: __list_options__. You can select an option, summarize an option, or search for something else.