

---

# GRILLBot: A flexible conversational agent for solving complex real-world tasks

---

**Carlos Gemmell**

University of Glasgow

c.gemmell.1@research.gla.ac.uk

**Sophie Fischer**

University of Glasgow

sophie.fischer@glasgow.ac.uk

**Iain Mackie**

University of Glasgow

i.mackie.1@research.gla.ac.uk

**Paul Owoicho**

University of Glasgow

p.owoicho.1@research.gla.ac.uk

**Federico Rossetto**

University of Glasgow

f.rossetto.1@research.gla.ac.uk

**Jeff Dalton**

University of Glasgow

jeff.dalton@glasgow.ac.uk

## Abstract

We present GRILLBot, a multi-modal task-oriented voice assistant to guide users through complex real-world tasks for the Alexa TaskBot Challenge. An effective TaskBot has to guide a user through a long and complex task, be engaging, and help solve problems along the way. GRILLBot achieves this in the domains of cooking and home improvement by helping search over a large task corpus with mixed-initiative and executing those tasks with neural dialogue management, knowledge grounded question answering, and web information extraction. To represent each task, we propose TaskGraphs as a dynamic graph unifying steps, requirements, and curated domain knowledge enabling detailed contextual explanations and adaptable task execution. Automatic linking of multi-modal elements helps the user navigate through the task and enrich the experience with helpful videos and images. Broad use of neural language models makes for flexible chit-chat, contextual intent parsing, and accurate task retrieval. Upon winning the competition, GRILLBot achieves a average rating in May (1st to 11th) of 3.86/5.0 and excels in longer conversations.

## 1 Introduction

In this paper, we describe GRILLBot, a task-oriented multi-modal conversational agent built as a research framework during the 2021/2022 Alexa Prize TaskBot challenge. GRILLBot focuses on the domains of cooking and home improvement and maintains deep conversations while walking a user through accomplishing a complex task. GRILLBot enables users to learn while they are performing the task at hand, handling chit-chat with humor and engaging the user with extra information.

GRILLBot is built as a platform-agnostic research framework published as OAT (Open Assistant Toolkit)<sup>1</sup>. Web-scale offline processing and augmentation allow for a broad spectrum of tasks within the domains. Although the system currently focuses on these specific domains to suit the scope of the competition, it is data-driven and open to broader task coverage. This large set of curated tasks is made available to the user when they invoke GRILLBot. During a live interaction, GRILLBot

---

<sup>1</sup><https://github.com/grill-lab/OAT>

leverages a phased design to the overall conversation structure to first help the user choose a task, and then help them execute it. In the final month of finals, we achieve an average rating of 3.86/5.0. We analyze in more detail the ratings in section 6.

We develop TaskGraph, a novel dynamic representation of steps, requirements, and extra elements to walk users through intricate tasks. TaskGraph enables fluid and adaptable conversations by providing a structured abstraction of the task. It allows GRILLBot to take the initiative, or for the user to make decisions on the task at hand.

The TaskBot challenge is the first Alexa challenge to introduce multi-modality as a core part of the user experience. Complex tasks in the domains of cooking and home improvement benefit greatly from suitable visual guides. GRILLBot takes full advantage of this new modality by providing “How-To” video guides for complex instructions. We also augment existing tasks with semantically relevant images for tasks and steps that do not have any.

2

## 1.1 Dialogue Schema

GRILLBot represents each user conversation as a finite set of phases that the user traverses as they go through the task. Depending on the phase, different kinds of system behavior are exposed to the user. Each phase represents a specific user goal and allows us to model different interaction schemes with phase-specific functionality.

This approach also provides the flexibility to handle complex conversational flows. For example, users can explore various recipes by iteratively selecting recipes to explore. We also encode the phase information as input to the Neural Decision Parser (Section 5.2) as part of the conversational state. An example conversation is shown in Fig. 1 where the phases are as follows:

- **Domain:** Bot introduction and selection of one of the domains based on the user request.
- **Planning:** Search for a suitable task based on the user request. In this phase, we recommend a task or guide the user through the selection process.
- **Validation:** Choice confirmation. We show the task specifics to the user to confirm their choice. This includes any required materials or tools that the user will need.
- **Execution:** This is the primary phase, where the user is guided through the different task instructions, until the end of the task.
- **Farewell:** Exit phase, where we congratulate the user.

## 1.2 Multimodality and UI design

Input and output modalities such as touch, screen, and voice play a crucial role in effective communication for complex tasks. GRILLBot adapts its behavior based on the available modalities to tailor the user experience to the supported devices: voice-based (e.g. Echo Dot) and screen devices (e.g. Echo Show).

**Voice UI** Presentation of content is important, as every unnecessary or unnatural turn can cause early conversation dropout. Within each conversation phase, we A/B tested prompts to achieve the optimal speech response to let the user feel like the system can understand them and appropriately respond to their queries.

During the conversation with the user, GRILLBot maintains a fine balance between giving the user enough information about the step and not overloading the user. Especially when using a headless device, the Alexa voice response should be more concise, as the user cannot hear and remember too much content. An issue we faced during the competition was discoverability, as the user was often not aware of many key features that were available. Therefore, we carefully crafted prompts and system initiative to instruct the user. We added these to both the speech and the visual UI to teach to the user the platform capabilities.

---

<sup>2</sup><https://github.com/grill-lab/OAT>

DOMAIN	TaskBot	Hi, this is an Alexa Prize TaskBot. I know about cooking and home improvement. What can I help you with?
	User	I don't know, help me cook something
PLANNING	TaskBot	I love roasted anything. Asparagus and Sweet Potatoes are my favourite. It's fast, easy, and a big bang of deliciousness for very little effort. What's your favorite thing to cook?
	User	I love smoked meat
	TaskBot	So, for "smoked meat", I found three great matches. The first one is: Smoked Meat Sandwich. The second is: K I S S Smoked Chicken. And, finally: Mustard Molasses Smoked Ribs. Which would you like?
		⋮
	TaskBot	You chose Mustard Molasses Smoked Ribs. It's got a rating of 4.0 stars and has 17 steps. Do you want to hear the ingredients?
VALIDATION	User	Yes!
	TaskBot	Okay, for starters, you'll need 1 jar Creole or whole-grain mustard. Second, 3 ounces unsulfured molasses. And, third, 1 cup lager-style beer. Just so you know, I'll tell you the ingredients three at a time.
	User	Actually, just start with the recipe
EXECUTION	TaskBot	Let's get started with Mustard Molasses Smoked Ribs. So, for Step 1 out of 17, light a grill or smoker to medium heat (about 300°F).
		⋮
	TaskBot	Step 4. Season ribs with salt and pepper.
	User	How much kosher salt do i need?
	TaskBot	Two teaspoons
	User	Oh ok got it, what's next?
	TaskBot	Step 5. Place 2 sheets of foil on a work surface and set a rack of ribs in the center; pull the edges of the foil up around ribs.
		⋮
	TaskBot	Step 17. Brush ribs with the reduced mop sauce, transfer to a platter and serve.
	User	Ok, I think it is ready!
FAREWELL	TaskBot	Great job, your recipe is complete and you managed to make food with a robot, how awesome! I had lots of fun cooking with you, hope you had too!

Figure 1: Example conversation with phases for each utterance. The user utterances are from a team member in the interest of preserving user privacy.

**Screen UI** We use the visual channels to improve communication of intricate steps, as well as keep user engagement throughout the process. When a screen is available, we incorporate custom multimodal elements along with the system speech. This includes images, buttons, rendered text, videos as well as helpful prompts for the user to try out.

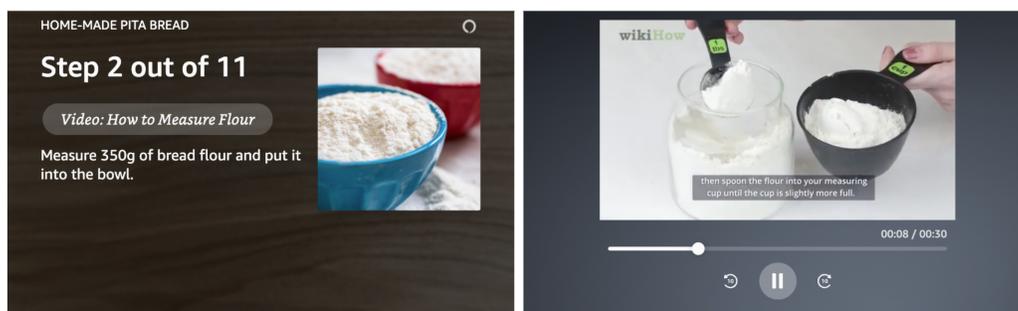


Figure 2: Example recipe page during execution (Left). Automatically linked "How-to" video (Right)

## 2 System overview

GRILLBot is designed to provide a platform-specific interface (Alexa Bridge) that abstracts the multi-modal input and output from the core conversational systems. The system allows the design of rich conversational flows that can easily access conversational state, external modules, and data. When the system receives user input through a front-end, the client-specific (Alexa Bridge) interface provides a standard input to the system (flows 1 and 2). At its core, the Orchestrator manages the application logic to process the user utterance. This includes managing stateful attributes and calling

the appropriate stateless components to form the system response. Multiple phase policies allow us to isolate specific system behavior depending on the progress through the conversation. The policy can call external modules many times based on conversation design and module responses. The Orchestrator loads the previous conversation state that can be used as context (flow 3). The policy, which encodes the conversational flow, calls external modules (flows 4 and 5) - named Neural Functionalities and Functionalities - which provide stateless responses (i.e. task search, task QA, or policy decisions with the Neural Decision Parser). Finally, the policy decides what utterance or multi-modal object to return to the client (flows 6 and 7).

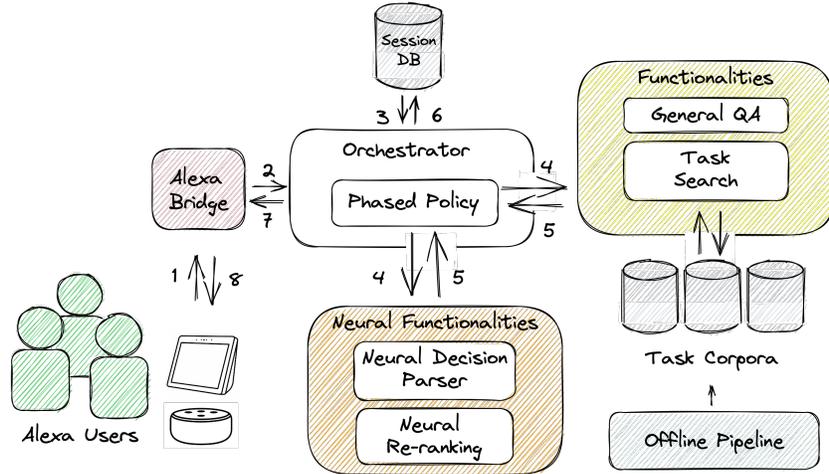


Figure 3: GRILLBot Online System Architecture. Numbers outline the order of operations.

GRILLBot’s online pipeline is decomposed into four micro-services. First, we classify the user query into a topic domain. We then use a Neural Decision Parser to direct internal state transitions handling search, navigation, and question answering based on the user turns during the conversation. Furthermore, GRILLBot uses a curated search service to retrieve and rerank matching tasks from the system’s task corpora. Finally, GRILLBot helps users complete their tasks through question-answering capabilities grounded in the domain and task knowledge.

Communication between components is performed using structured API calls defined in Google Protocol Buffers<sup>3</sup> and using the gRPC framework<sup>4</sup>.

### 3 Task Representation - TaskGraph

We develop a new representation for the task structure to enable complex flows of execution. The *TaskGraph* is a directed acyclic graph that represents a model of the actions and information dependencies that the system uses to aid the user in performing the task. Information is represented with heterogeneous nodes, each with a specific role:

- **Steps:** They represent an instruction to be given to the user, including a shorter summary and a longer description. They also contain all the visual information required to visualize the step.
- **Requirements:** They represent Tools and Ingredients that are needed to perform the task and are usually linked to specific steps.
- **Conditions:** They represent yes/no gates that require external information which is dynamically resolved during execution. This enables the system to adapt based on user behavior.
- **Logic:** They represent logical operations. These can be used in conjunction with other nodes to enable compact dependencies and a smoother flow of execution. We currently support  $\wedge$ ,  $\vee$  and  $\neg$  operations.

<sup>3</sup><https://developers.google.com/protocol-buffers>

<sup>4</sup><https://grpc.io>

- **Actions:** Nodes representing actions taken by the system. This could be operations like setting a timer or adding items to a list, but they can also represent live retrieval of domain knowledge.
- **Extra Information:** Nodes representing domain/task-specific knowledge like tips or fun facts that can enrich the user experience during the execution of the task.

Combining all these nodes, we can obtain adaptive interactions where system-initiative can allow the system to adapt to the user’s needs.

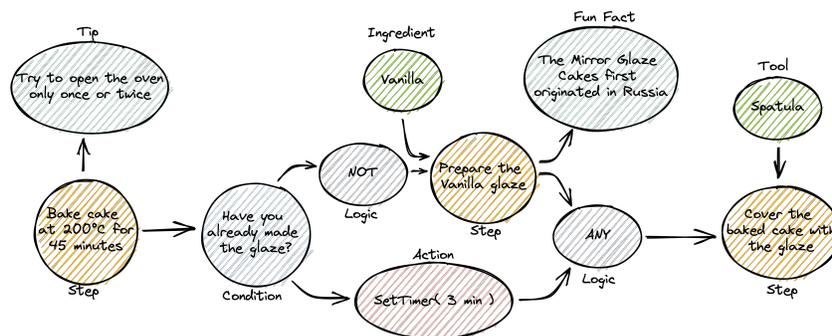


Figure 4: Example TaskGraph for baking a cake.

Fig. 4 shows how the graph representation enables providing extra information when discussing each step. Moreover, we can adapt instructions based on the user’s needs. For example, we can add instructions for preparing a cake glaze, if the user wishes to do so. We perform actions, like automatically starting a timer to wait for the user cake to bake. The linked ingredients and tools allow us to guide the user during the execution of the step. The logical node ANY allows for a compact representation collecting parallel and mutually exclusive dependencies. The logical node NOT instead allows the Conditional node to act like an IF-switch.

A benefit of abstracting the task as a TaskGraph is that it allows flexibility in system planning of the flow of the conversation. It is manually inspectable and editable, giving the possibility to correct issues like long instructions, and it allows optimization for popular or challenging tasks for the best user experience possible.

## 4 Offline TaskGraph Curation

An essential aspect of the system is providing relevant and high-quality TaskGraphs given a user’s goals. For example, if a user asks, “I want to cook a healthy dairy-free meal for a dinner party based around scallops“, the system must access TaskGraphs that will assist the user in achieving this task. This requires the system to process rich and executable TaskGraphs offline with enough scale to cover most user needs. Fig. 5 shows the custom TaskGraph corpus processing pipeline, which shifts heavy processing stages and data enrichment to a specific offline pipeline. This reduces the online processing required while allowing for computationally heavy enrichment of our sources.

For each domain, we rely on domain experts to identify high-quality seed websites. We use Common Crawl to download the raw HTML for target domains and develop website-specific wrappers to extract semi-structured information about each task, i.e. title, author, description, ingredients, images, steps, ratings, videos, etc. We use multiple crawls (i.e. CC-MAIN-2021-[21,17,10,04]) to ensure adequate coverage. We also extract other metadata for search or clustering (tags, links, etc.) and additional information for QA (infoboxes, FAQs, etc.).

The next stage of the offline pipeline takes the semi-structured information extracted and synthesizes executable TaskGraphs. For example, we create multi-modal task nodes and connections from previously linear steps. If there is too much text within certain task nodes, we will create a summary and a detailed description that can be accessed by users who require additional context. Finally, we filter out poor quality TaskGraphs that do not contain the ingredients for cooking, too many steps, the task is out of the domain, etc. After the creation of these TaskGraphs, we run a pipeline that links the ingredients and tools to each step in which they are required. This is done by matching normalized

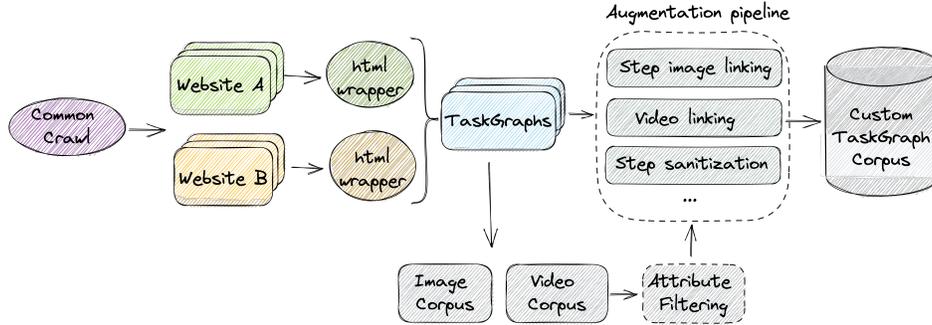


Figure 5: GRILLBot Offline Pipeline including multiple types of parallel augmentations.

noun chunks from the step and requirement text. This normalization is done using lemmatization of the root nouns. This enables us to improve the representation of the task with more granular information, that we can then provide to the user during the execution of each step.

Lastly, we store the collection of TaskGraphs in several indexes to allow complex queries to be run over the corpus. This includes storing TaskGraphs within a Lucene index built with Pyserini<sup>5</sup> to allow fielded and sparse retrieval (BM25 [Robertson and Walker, 1994] and BM25+RM3 [Abdul-Jaleel et al., 2004]). We also index specific text fields (title, description, and ingredients) offline for dense retrieval (ANCE [Xiong et al., 2020] and ColBERT [Khattab and Zaharia, 2020]). TaskGraph offline processing and index generation allows us to offload long and complex operations to enrich our corpus as well as provide an efficient online interface.

After pre-processing and filtering, our task corpus contains 165,677 TaskGraphs encompassing home improvement and cooking. To assess coverage we sample user queries from our live system and annotate the results. We find that over 95% of user tasks are covered by this corpus.

## 5 Online System

In this section, we describe the core GRILLBot components to naturally walk a user through completing a complex task. These are isolated functionalities that are strung together by the Orchestrator. This component is in charge of defining the behavior of the system, using the functionalities as tools. It contains most of the prompts and logic when dealing with the user.

### 5.1 Domain Classification

One of the initial components of our system is a simple Naive Bayes classifier that helps us determine which domain the user request belongs to. At the beginning of the conversation, the classifier determines which Task-Index to search from, and during QA interactions, it determines if the user request is relevant to our current scope.

We define six possible domains: **Cooking**, **DIY**, **Medical**, **Financial**, **Legal** and **Undefined**. To train the model we have collected Reddit conversations [Baumgartner et al., 2020] from the past years from thematic sub-reddits. We also post-process the output distribution filtering based on a probability threshold and an entropy threshold to determine the model confidence in the domain prediction. In cases of strong uncertainty, we prefer the **Undefined** domain.

### 5.2 Neural Decision Parser

One of the strengths of GRILLBot is its ability to understand user commands in natural language. This is enabled by a unified contextual semantic parser, which takes natural language representations of current and past state and generates actions in form of the custom GRILLBot Domain Specific Language (DSL). These generated arguments are unbounded and supply the task sub-components with parsed knowledge relating the context to the current utterance. Fig. 6 shows an overview of our end-to-end contextual semantic parser used in various states throughout the system to direct flow.

<sup>5</sup><https://github.com/castorini/pyserini>

**State Transitions as a DSL** We define our state transition DSL to include all system actions. This DSL outlines a parametrized global command set that is understood throughout the system to derive what actions or external APIs should be called, and what the response utterance should be.

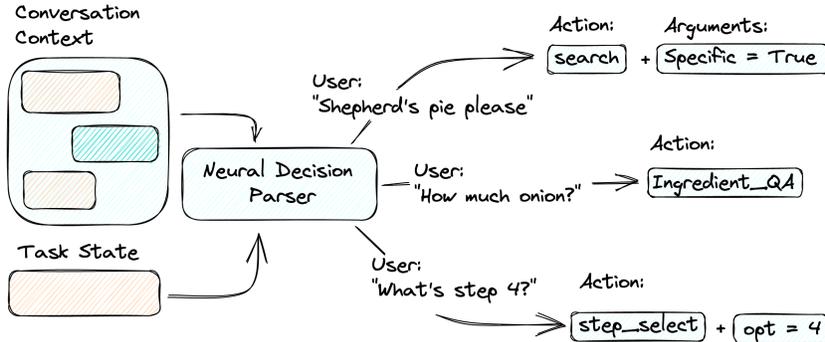


Figure 6: GRILLBot Neural Decision Parser architecture

**Implementation Details** We use a single adapted T5 [Raffel et al., 2020] model to flexibly generate an agent action based on the previous state and provided context. This allows for advanced language capability across all areas of the system seamlessly including coreference resolution, search parameterization, domain classification, and state prediction.

We train our system by annotating real user conversations with the appropriate function calls to be generated. This includes the intent and associated arguments to the function call for turns across various stages of the conversation. Each sample consists of the previous system utterance and the current user utterance. This ensures context is available to the model to resolve ambiguous utterances. Many of the simpler intents ("more details", "next step", "yes", etc) are easily captured by our parser. However, the majority of our training samples consist of complex contextual utterances resolving coreference in task selection. This is because distinguishing between search and question answering is a very nuance problem.

When training, we find the model size to be an important factor to generalize to new samples. Table 2 shows aggregated intent and argument exact match accuracy increasing with model size. Notably, we find complex arguments like timer spans and coreference resolution benefit most from the deeper language understanding of bigger models.

Ranking Method	dev	test
BM25	0.190	0.234
ANCE	0.274	0.196
BERT-title	0.561	0.532
L2R	0.658	0.534

(a) Table 1: Task retrieval evaluation aggregated over the domains of cooking and home improvement.

Neural Decision Parser	dev	test
T5 small	0.613	0.622
T5 base	0.781	0.828
T5 large (ours)	0.838	0.863

(b) Table 2: Neural Decision Parser performance on combined intent and attribute accuracy.

### 5.3 Federated TaskGraph Retrieval

We define the task-specific search as the process of retrieving a ranked list of TaskGraphs ( $[T_1, T_2, \dots, T_N]$ ) from a corpus ( $C_T$ ) given an information need ( $Q$ ). These highly ranked results can then be presented to the user for selection (“(1) Pizza Bufalina by Kitchen Stories and has a rating of 4.7 stars; (2)...”). Precision at high ranks (i.e. top 3) is particularly important given restricted voice and conversational paradigm, similar to the work in the TREC Conversational Assistance Track [Dalton et al., 2019].

If GRILLBot detects a search intent in the live system, a multi-pass retrieval pipeline performs federated retrieval over multiple corpora. This includes our custom corpus (with Lucene-based sparse and dense indexing) and several third-party APIs. This allows for standard sparse retrieval methods (BM25 [Robertson and Walker, 1994], query expansion, [Abdul-Jaleel et al., 2004]), dense retrieval

methods (ColBERT [Khattab and Zaharia, 2020] and ANCE [Xiong et al., 2020]). We find combining the returned TaskGraphs from different indexes increases recall.

We leverage a learning-to-rank to combine various features of TaskGraphs encompassing query similarity, quality, and usability. These include text-based relevance features of query similarity to the fields within the task, BM25 scores, T5 scores [Nogueira et al., 2020], Jaccard, and Boolean features. Additionally, we include query-independent quality features, including source, author, popularity, and user rating. Furthermore, we add usability features such as images, a manageable number of tools and ingredients, and a reasonable number of words in each step. We use weak supervision signals from industry search engines to generate training and validation data. Where coordinate ascent, a gradient-based listwise method, is used to optimize mean NDCG@3 directly. Finally, findings from log analysis are incorporated to make any final parameter changes based on specific failures.

## 5.4 Multi-media Augmentation

Visual information often plays a key role in user task success. Using the screen interface, GRILLBot complements voice instructions with multi-modal elements such as videos and images.

**Step Image Augmentations** When selecting or walking a user through a task, images allow the user to anchor their actions in visible examples. In situations where step images are not available, GRILLBot augments the voice instruction with relevant steps by retrieving similar ones from other tasks. Our image corpus is derived from all images present in our task corpus resulting in over 101,241 images to search over. We derive a semantic similarity measure between images and the step text through the use of CLIP Radford et al. [2021] and compute the cosine similarity between single embedding representations of each modality. In the case similarity between the step text and the best image result no be higher than a predefined threshold we default to the thumbnail image.

Image augmentation is most relevant in the cooking domain where steps in recipes are often sparsely accompanied by images. Notably when looking for a specific result like “caramelizing onions” using an accompanying image informs the user of what color they should be if they don’t know the technique.

**"How-to" Video Augmentations** We target technique videos to help users with particularly complex steps. These videos target cooking and DIY methods that are reusable across multiple different tasks such as “How to dice onions”. GRILLBot uses a corpus of these technique videos and provides suitable videos through voice prompting and screen buttons.

Importantly, not all TaskGraphs we obtain through our offline curation process have videos natively in their instructions. Out of 18,455 WikiHow articles we scraped, only 10% of them have a video associated resulting in 2,140 videos. Once filtering this WikiHow corpus to short 1-minute videos from reliable sources, our video corpus consists of 733 technique videos. To decide whether a suitable video in our corpus exists for a given step, we employ a curated set of action verbs (i.e. “dice”, “cut”, etc) as a preliminary filter, and extract spans using spaCy Honnibal and Montani [2017]. Similar to Image retrieval, we embed the extracted span as a query and rank the titles of each “How-To” video through cosine similarity of the sentence embedding.

## 5.5 Question Answering

When guiding the user through a task, questions often extend beyond general trivia into context-dependent queries to clarify instructions, requests previously seen information, or expand on a topic within the domain. Furthermore, their utterances might be loosely related to the task at hand and derive into a more chit-chat-focused dialogue.

GRILLBot gracefully handles many different kinds of interactions that are not directly related to driving forward through the task. Specifically, any nonstandard user utterance is classified into 9 categories show in table 2

Each of these types of questions is handled by an appropriate QA system. Each system is responsible for aggregating the required context to answer the question. This is especially important in highly contextual situations like a task question that will depend entirely on stateful information. Should

Ingredient question	“How much oil do I need?”
Current task question	“How hot was the oven again?”
Step question	“How many steps are left?”
General cooking or DIY question	“How does wood glue work?”
Other domain question	“What’s the capital of Paris?”
Chit-chat	“What’s your favorite recipe?”
System capabilities question	“How do I set a timer?”
Current viewing options question	“Can you compare these recipes?”
Ingredient substitution	“I’m dairy-free, is oat milk alright?”

Table 2: Question categories

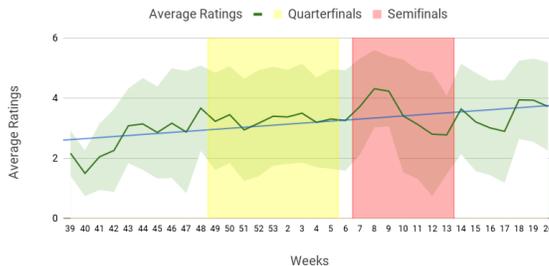
the question be non-sensical, it is left to each system to handle fallback gracefully to inform the user their question cannot be answered.

For system capabilities, out of domain questions, and chit-chat, we employ neural language models to synthesize fluent responses with a small paragraph of context appended as a prompt. These models are used in a Zero-Shot approach to retain maximal fluency. When synthesizing an answer we encourage the model through the prompt to always bring the conversation back to the domains of cooking and home improvement. This is especially important when the language model does not know the answer to a question and needs to both inform the user of this and make them aware of their capabilities.

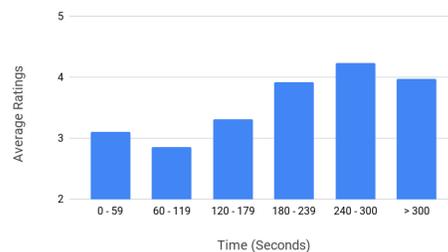
Our approach to task-specific QA (ingredient, task, and step questions) involves creating a natural language (NL) representation of the current state and processing it jointly with the user query to derive the answer. This NL representation includes the task requirements and task steps. We use zero-shot UnifiedQA [Khashabi et al., 2020] to autoregressively generate an answer.

We ensure fluent and accurate generated sequences through constrained generation by restricting model output vocabulary  $V$  by only allowing generated tokens to be present in the context  $C$  similar to an extractive QA system. The query  $Q$  also forms part of the context. Answers are abstractive since users can ask “yes/no” questions that might not be in the context. These additional tokens form part of a token whitelist set  $S$ .  $V$  is then derived from the union of  $C$ ,  $Q$ , and  $S$ . General QA allows the system to answer different question types: general domain questions and conversion/table lookups. We perform a call to the Amazon Evi API when we encounter these types of domain questions.

## 6 Analysis



(a) Average ratings throughout the competition



(b) Average ratings based on conversation length

During the competition, we observed user interactions and rating quality to measure system performance. We have been able to optimize based on this, and improve the received average user ratings. Fig. 7a shows the improving trend of our the ratings throughout the competition. During weens 7 and 9 we released preference elicitation which we describe in Section 6.1. We observe a marked increase in user ratings during this time with conversation analysis highlighting users able to advance conversationally through to deeper sections of the phased journey.

Fig. 7b shows the average ratings by conversation length. Conversations under 60 seconds usually came from a user that started the bot by mistake, resulting in a bad rating. For conversations longer than 1 minute, the ratings increase proportionally with the conversation duration. This trend continues up until the conversation duration of 5 minutes when ratings stabilize around 3.9-4.0.

This is reflected across the ratings of screen devices at the end of Semi-Finals. Average user ratings using the multi-modal screen device are 0.26 higher (3.62 vs 3.36).

### 6.1 Preference Elicitation

Interactions with the initial iterations of GRILLBot were heavily user-led. Typically, a user would query the system for a task they wished to perform (i.e. “help me make pizza”), select one of the options presented to them (i.e. “the first one”), and then issue navigation commands (i.e. “next”, “previous”, “repeat”) until they get to the end of the interaction.

TaskBot	Hi, I'm GRILLBot. What do you want to do?	TaskBot	Hi, I'm GRILLBot. I know about cooking, home improvement, arts & crafts. What do you want to do today?
User	I want to cook	User	how to write a movie review
TaskBot	I understood "I wanna cook" say cancel if you want to start over. Top search results are: first, I wanna cook by food ...	TaskBot	Finding recipes and walking you through arts and crafts projects is what I do best. I don't yet know about other areas though. I'd love to help you out with something like, how do I remove a stain from a carpet.
User	no, stop	User	how to write a movie review
		TaskBot	Discussing things other than recipes or do it yourself tasks isn't really my strong suit. What we can do is find a recipe or project to do and walk through it together! Some of my favourite are: 'making carrot cake', or 'giving a wooden table new life with some varnish'. Let me know what you'd like.
		User	stop

Figure 8: Typical conversations before (left) and after (right) preference elicitation.

While this linear flow works well sometimes - especially with users who have an intuition for how a system like GRILLBot should work - many users do not have a clear objective or are simply unaware of the system’s capabilities.

This phenomenon manifests in the form of “vague queries”; users ask GRILLBot to help them “cook something” or “recommend good recipes to try”. Without additional context, such queries return poor search results and contribute to user frustration and early abandonment. In fact, in the early stages of the competition, 28.8% of users who rated their interaction with GRILLBot two stars or lower did not make it past the planning phase of the conversation. Moreover, only 5.8% of these users finished their tasks, highlighting the need for an intuitive interaction flow that leaves users feeling understood.

Based on previous work [Radlinski et al., 2019, Horvitz, 1999], we mitigate this through the notion of preference elicitation. On account of the Neural Decision Parser, GRILLBot recognizes when a user needs guidance in defining an objective. With this, GRILLBot initiates a sub-dialogue to extract the user’s preferences and makes meaningful recommendations based on those preferences. After implementing preference elicitation, we see a 12.96% improvement in overall user ratings and satisfaction. Particularly, fewer users drop off in the planning step, highlighting GRILLBot’s increased effectiveness at helping users decide what to do. Additionally, feedback from testers stresses the naturalness of the preference elicitation sub-dialog as a strong point for the system.

## 7 Conclusion

We present GRILLBot, a multi-modal task-oriented voice assistant to guide users through complex real-world tasks in the domains of cooking and home improvement for the Alexa TaskBot Challenge. Our system leverages a new task representation, TaskGraphs, as a dynamic graph unifying steps, requirements, and curated domain knowledge to enable contextual question answering and detailed explanations. We leverage automatically linked multi-modal elements to help the user navigate through the task. Our Neural Decision Parser enables flexible navigation by jointly encoding stateful information with the conversation history and engages with personality and knowledge grounded question answering. Our system makes a step towards multi-modal agents that guide and engage users through achieving real-world tasks.

## 8 Acknowledgments

The authors would like to thank the following researchers for their support and collaboration: Alessandro Spegiorin, Ivan Sekulić, Gijs Hendriksen, and Alexander Frummet. Additionally, this work is supported by a Turing AI Acceleration Fellowship from the Engineering and Physical Sciences Research Council, grant number EP/V025708/1.

## References

- N. Abdul-Jaleel, J. Allan, W. B. Croft, F. Diaz, L. Larkey, X. Li, M. D. Smucker, and C. Wade. Umass at trec 2004: Novelty and hard. *Computer Science Department Faculty Publication Series*, page 189, 2004.
- J. Baumgartner, S. Zannettou, B. Keegan, M. Squire, and J. Blackburn. The pushshift reddit dataset. *CoRR*, abs/2001.08435, 2020. URL <https://arxiv.org/abs/2001.08435>.
- J. Dalton, C. Xiong, and J. Callan. CAsT 2019: The Conversational Assistance Track Overview. page 10, 2019.
- M. Honnibal and I. Montani. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. 2017.
- E. Horvitz. Mixed-initiative interaction. *IEEE Intelligent Systems*, pages 14–24, September 1999. URL <https://www.microsoft.com/en-us/research/publication/mixed-initiative-interaction/>.
- D. Khashabi, S. Min, T. Khot, A. Sabharwal, O. Tafjord, P. Clark, and H. Hajishirzi. Unifiedqa: Crossing format boundaries with a single qa system. *arXiv preprint arXiv:2005.00700*, 2020.
- O. Khattab and M. Zaharia. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proc. of SIGIR*, pages 39–48, 2020.
- R. Nogueira, Z. Jiang, R. Pradeep, and J. Lin. Document ranking with a pretrained sequence-to-sequence model. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 708–718, 2020.
- A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021.
- F. Radlinski, K. Balog, B. Byrne, and K. Krishnamoorthi. Coached conversational preference elicitation: A case study in understanding movie preferences. In *Proceedings of the 20th Annual SIGDial Meeting on Discourse and Dialogue*, pages 353–360, Stockholm, Sweden, Sept. 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-5941. URL <https://aclanthology.org/W19-5941>.
- C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *arXiv:1910.10683 [cs, stat]*, July 2020. URL <http://arxiv.org/abs/1910.10683>. arXiv: 1910.10683.
- S. E. Robertson and S. Walker. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *SIGIR'94*, pages 232–241. Springer, 1994.
- L. Xiong, C. Xiong, Y. Li, K.-F. Tang, J. Liu, P. N. Bennett, J. Ahmed, and A. Overwijk. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *International Conference on Learning Representations*, 2020.