Advancing conversational task assistance: The Second Alexa Prize TaskBot Challenge

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**

Reza Ghanadan

Abstract

Since its inception in 2016, the Alexa Prize program has enabled hundreds of university students and faculty to explore and compete in the development of conversational agents through the SocialBot Grand Challenge, whose goal is to build agents capable of conversing coherently and engagingly with humans on popular topics. As conversational agents attempt to assist users with increasingly complex tasks, new conversational AI techniques and evaluation platforms are needed. The Alexa Prize TaskBot Challenge, now in its second year, introduced the requirements of interactively assisting humans with real-world tasks, while making use of both voice and visual modalities. This challenge requires the TaskBots to identify and understand the user's need, identify and integrate task and domain knowledge into the interaction, and develop new ways of engaging the user without distracting them from the task at hand, among other challenges. This paper provides an overview of the second TaskBot challenge, in which both new and returning teams participated. We describe the infrastructure support and the new models provided to the teams with the CoBot Toolkit. We then summarize the approaches the participating teams took to address research challenges, including changes and improvements from the previous year. Finally, we analyze the performance of the competing TaskBots and discuss some of the lessons learned.

1 Introduction

In the last several years, conversational assistants have become a popular way for humans to access information. As humans expect to interact with such agents by voice, the conversational assistants must be able to support a broader range of increasingly complex tasks. The Alexa Prize[1] is an Amazon

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

2nd Proceedings of Alexa Prize TaskBot Challenge (Alexa Prize 2022-23).

Alexa sponsored program that has enabled hundreds of university students and faculty to compete in advancing the state-of-the-art in conversational AI. Since 2016, the Alexa Prize program has hosted a competition among universities across the world to compete in creating the best *SocialBot*, i.e., an Alexa skill that can engage in extended open-domain dialog with users on popular topics and current events [21]. The SimBot Challenge, started in 2022, pushes the boundaries of embodied conversational AI by challenging teams to build SimBots that users can instruct to complete tasks in a simulated 3D environment ([40]). One of the key advantages of the program is that it enables university teams to rapidly test and iterate on their approaches through testing with real world users at scale through Alexa.

Since 2021, the TaskBot Challenge has engaged teams in building conversational assistants that can assist users to complete complex tasks such as recipes or Do It Yourself (DIY) projects [19]. In many cases, users search the internet to find potential activities, such as how to cook a specific dish or tutorials on how to complete a DIY project, but these experiences are neither interactive nor conversational. To advance the state of the art in interactive conversational assistance we introduced a new Alexa Prize challenge to develop a TaskBot, i.e., a conversational agent that can interactively assist users in completing complex real-world tasks. Just as in the SocialBot challenge, the TaskBots must maintain helpful and engaging interactions with the user. However, the purpose of a TaskBot is not merely to converse, but to actively assist the user in formulating and accomplishing their task. This setting is quite different from existing task-oriented assistants, where the user typically provides information to the task assistant to let the machine perform a task (e.g., booking a hotel, buying flight tickets). A TaskBot, instead, is meant to provide the humans with both accurate information and assistance about a task that is executed by them. We refer to this setting as Conversational Task Assistance. Unlike the web search experience, a TaskBot is interactive, enabling the users to find a task with respect to their needs by conversing with them; once a task is found, a TaskBot agent leads them through the task and helps in answering questions and addressing issues along the way. A key focus of the challenge is that the competing TaskBots are required to be fully multimodal; that is they help the users not just through a spoken conversation but through an integrated combination of spoken and graphical interaction.

Figure 1: The TaskBot Challenge 2 Displayed on an Echo Show 8.

For the TaskBot challenge, the participants build conversational agents on top of the multimodal Alexa service for two domains: Cooking and Home Improvement (or DIY). These present different challenges that require scientific advancements for a conversational system. For example, the TaskBot should help the user find the right task according to their needs; provide information about specific aspects of the task (e.g., ingredients or tools/materials); answer questions the user may ask about the task itself (e.g., how to use a specific tool); identify, structure, and integrate domain knowledge into the interaction; and generate responses for both voice and visual modalities. These technical challenges required and inspired the participants to invent novel technologies in different areas, like information retrieval, question answering, dialog management, and multimodal interaction.

For the Second TaskBot Challenge, ten university teams were selected to compete. Among them five teams were returning competitors from the previous year of the TaskBot Challenge. The selection criteria included the expected innovation of the teams' proposals and expected ability to execute on the proposed plan to develop an engaging and helpful experience for Alexa users. The selected teams participated in both an online and on-site Bootcamp (in Seattle) which helped them onboard to the challenge while introducing them to resources and support. The Bootcamp was followed by the

development phase where the teams were guided towards creating a TaskBot that could be deployed live to real users as an Alexa Skill.

After extensive testing and certification, the TaskBots were launched to a cohort of Amazon employees, followed by a public launch in May 2023, at which time all US Alexa users could interact with the participating TaskBots. Upon making a request for task assistance, by saying, "Alexa, Let's Work Together," Alexa users were connected to one of the ten participating TaskBots. After exiting the interaction with the TaskBot, the user was prompted for a verbal rating: "How helpful was this TaskBot in assisting you?" followed by a task completion prompt: "Were you able to complete your task?" and an option to provide additional free-form feedback. After an initial feedback phase, Semifinals were held, to which eight teams advanced.

From the Semifinals, five teams successfully advanced to the Finals phase and ultimately competed for top honors in the closed door finals event held in September 2023. During the live finals event, judges and interactors extensively tested and evaluated the TaskBots for coherence, helpfulness, engagement, and overall quality of the experience. Judges received training ahead of time, which included a thorough understanding of the challenge and what makes an excellent TaskBot. New additions to this year's finals event included the live performance of the tasks described (with making guacamole and origami crafting as sample tasks) and direct interactions by the judges with each finalist TaskBot (see Figure 2). This forced teams to work with the actual execution of the task, including pauses, mistakes, and real-time interactions while the recipe was being made and the task was being performed.

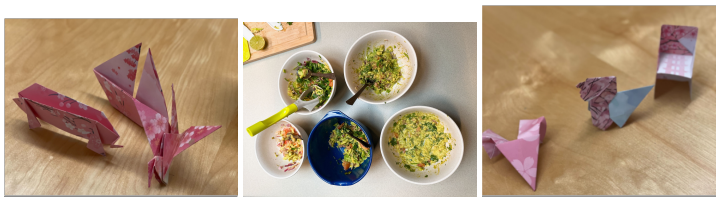

Figure 2: TaskBot Challenge 2 Live Finals Event Tasks.

The 2nd year of the TaskBot Challenge resulted in additional progress on the capabilities of the TaskBots, as illustrated by the participants' technical reports as well as the improvements in user satisfaction and engagement with the TaskBots as the competition progressed. In Section 2 Capabilities Provided to Teams we provide details on the capabilities provided to the teams. Section 3 Scientific Advancements summarizes the scientific advancements in the challenge both for participant teams and from the Alexa Prize team. The evaluation and performance of the TaskBots is detailed in Section 4 TaskBot Performance and overall insights gathered from the TaskBot Challenge and concluding remarks are in Section 5 Discussion and Conclusions.

2 Capabilities Provided to Teams

To facilitate research on TaskBot and advancing the science of multimodal conversational task assistance, the university teams were granted exclusive access to a range of Amazon Alexa resources, technologies, and experts in the science and engineering of Conversational AI systems. The following is an overview of the resources that were made available.

Specifically, to continue on the advances developed during the first TaskBot challenge, we released an updated version of the Conversational Bot Toolkit (CoBot) for developing conversational AI agents, as described below, together with supporting models and datasets, derived from the previous year of the TaskBot Challenge. Using the updated CoBot Toolkit, the participating teams were able to spend less time on engineering, and focus more on scientific advancements.

2.1 Conversational Bot Toolkit (CoBot)

We provided the TaskBot teams with CoBot [27], a conversational bot Toolkit in Python for natural language understanding and dialog management that has been used across both TaskBot and SocialBot competition tracks since 2018. CoBot includes a set of tools, libraries, and base models to help

develop and deploy open-domain or multi-domain conversational experiences through the Alexa Skills Kit [29] and Amazon AWS (see figure 4). CoBot's modular, extensible, and scalable design was developed based on learnings from previous Alexa Prize competitions and provides abstractions that enable the teams to focus more on scientific advances and reduce time invested into infrastructure, hosting, and scaling issues.

For the second TaskBot Challenge, we overhauled CoBot's deployment infrastructure to address major pain points in the continuous integration and continuous delivery (CI/CD) pipelines. CoBot is designed to give teams a CI/CD environment with which incremental code changes on bots are merged, delivered and released frequently and reliably to testing and production environments. This year, we minimized the iteration cycle time for each deployment by decoupling component deployments into separate pipelines that could be deployed independently, reducing the deployment time from 20 minutes to under 5 minutes. We estimate this optimization has saved 200 days worth of development time throughout TaskBot 2, enabling students to invest that time back into their research efforts. In addition, we also made significant changes in CoBot to support hosting large language models (LLMs) with upto 320B parameters, which is roughly 200 times greater than previously hosted in CoBot.

To support multimodal TaskBot development, we built on the previous set of Alexa Presentation Language (APL) [1] templates provided in CoBot to incorporate expanded customization capabilities, as described in Section 2.1.3 Multimodal Experience.

Lastly, we expanded the set of API offerings to include our latest generative neural models, a Cooking vs DIY Domain Classifier, a Harmful and Unauthorized Task classifier, and APIs for utterance rewriting and dialog evaluation, as described in Section 2.1.5 CoBot APIs.

Figure 3 illustrates the CoBot architecture. The modular design accelerates experimentation by enabling researchers to test and evaluate innovations in different components. We continue to invest in CoBot as our flagship toolkit for building open-domain dialog systems and constantly extend CoBot's capabilities to support the evolving demands of SocialBot, TaskBot, and other potential conversational bots in the future.

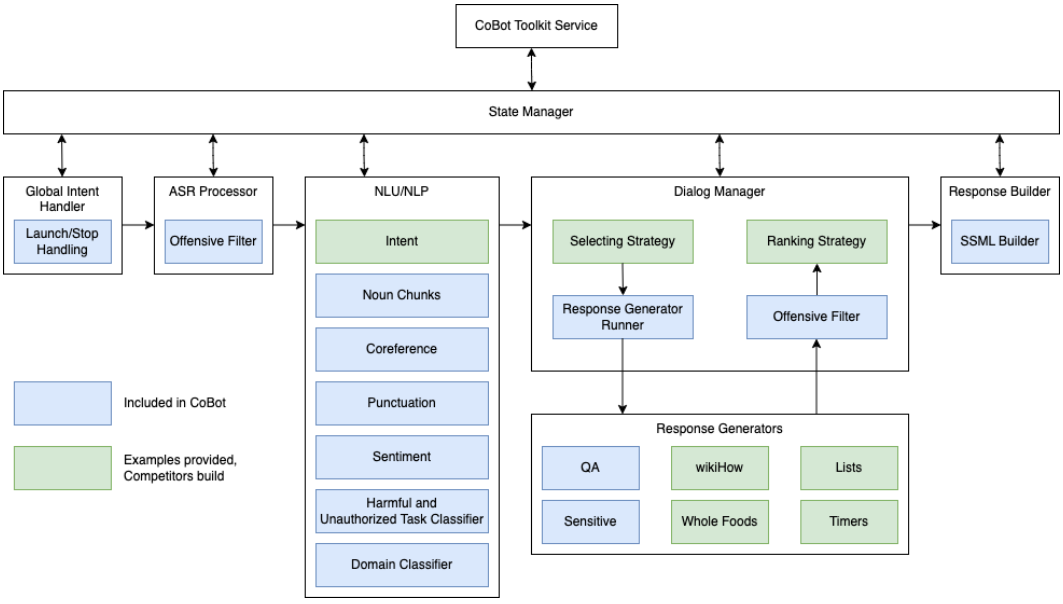

Figure 3: CoBot Architecture for TaskBot.

2.1.1 CoBot Deployment Infrastructure

CoBot is designed to give teams a CI/CD environment with which to rapidly deploy and iterate on bots. Figure 4 captures the system deployment architecture for a standard CoBot-based bot. CoBot uses AWS Lambda as a server-less infrastructure to host local modules. The Lambda is the endpoint of an Alexa skill and receives requests from the skill. CoBot uses Amazon Elastic Container Service

(Amazon ECS) and Docker to deploy and host bigger and long-running remote Docker modules. The module services sit behind Amazon Application Load Balancers (ALB). CoBot points the Lambda to the ALBs so that the bot can send requests from Lambda to remote Docker modules.

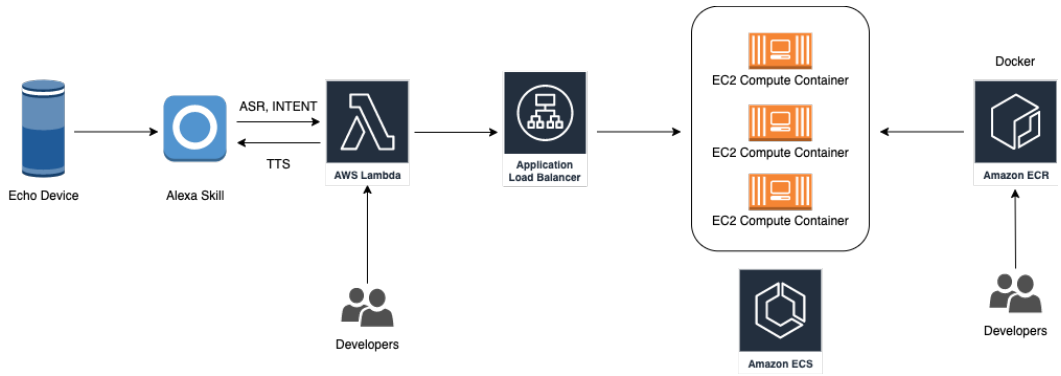

Figure 4: CoBot System Architecture.

CoBot builds a continuous delivery pipeline for an AWS Lambda application with AWS CodePipeline. The Lambda CodePipeline will monitor the Lambda CodeCommit repository for new commits, build the Lambda, and deploy it with AWS CloudFormation.

New for TaskBot 2, CoBot now builds a separate AWS CodePipeline for each remote Docker module. The CodePipeline monitors a module's CodeCommit repository for new commits, uses AWS CodeBuild to create a Docker container image and push it into Amazon Elastic Container Registry (Amazon ECR), and uses AWS CloudFormation to deploy the container image to production on Amazon ECS. A Flask [20] application sits behind an Amazon ALB to provide the scalability and resiliency to handle traffic from Alexa Prize users. Figure 5 shows the new design for deployment pipelines for remote Docker modules and Lambda.

The decoupling of remote Docker module pipelines allows researchers to only deploy the changes they need, without having to redeploy all modules. In addition, it facilitates tailoring of the GPU instance type used for hosting GPU-based containers to be different for each remote Docker module. This is a major cost-saving win, since researchers no longer need to use the largest GPU instance type for every GPU-based remote Docker module.

2.1.2 Large Language Model Support in CoBot

Hosting large language models poses some unique challenges. Due to the large size of these models, they generally have slower inference speeds and take a long time to load into memory. These restrictions make it difficult to host a real-time, low-latency service that can scale up to serve high volumes of user traffic. We added special provisions in CoBot to facilitate hosting LLMs as remote Docker modules (as described in Section 2.1.1 CoBot Deployment Infrastructure).

Inference speed: We utilized a lightweight Flask-based server to host LLMs in CoBot, and the inference engine was built using the HuggingFace transformers library [49]. All models were converted to bfloat16 for inferencing. This reduced their memory footprint, resulting in faster inference per GPU, and also enabled the model to be split between fewer GPUs, providing further gains in latency. We also experimented with numerous EC2 types and determined that G5 instances[2] with NVIDIA A10 GPUs[3] provided the fastest inference speeds with relatively low costs. We shared an example implementation of hosting a 5 billion parameter version of Alexa Teacher Model [43] to help university teams incorporate all these optimizations.

Autoscaling: CoBot remote Docker modules provide built-in auto-scaling policies based on CPU and Memory utilization. However, while hosting LLMs these policies proved ineffective for the following two reasons:

[2] `https://aws.amazon.com/ec2/instance-types/g5/`
[3] `https://www.nvidia.com/en-us/data-center/products/a10-gpu/`

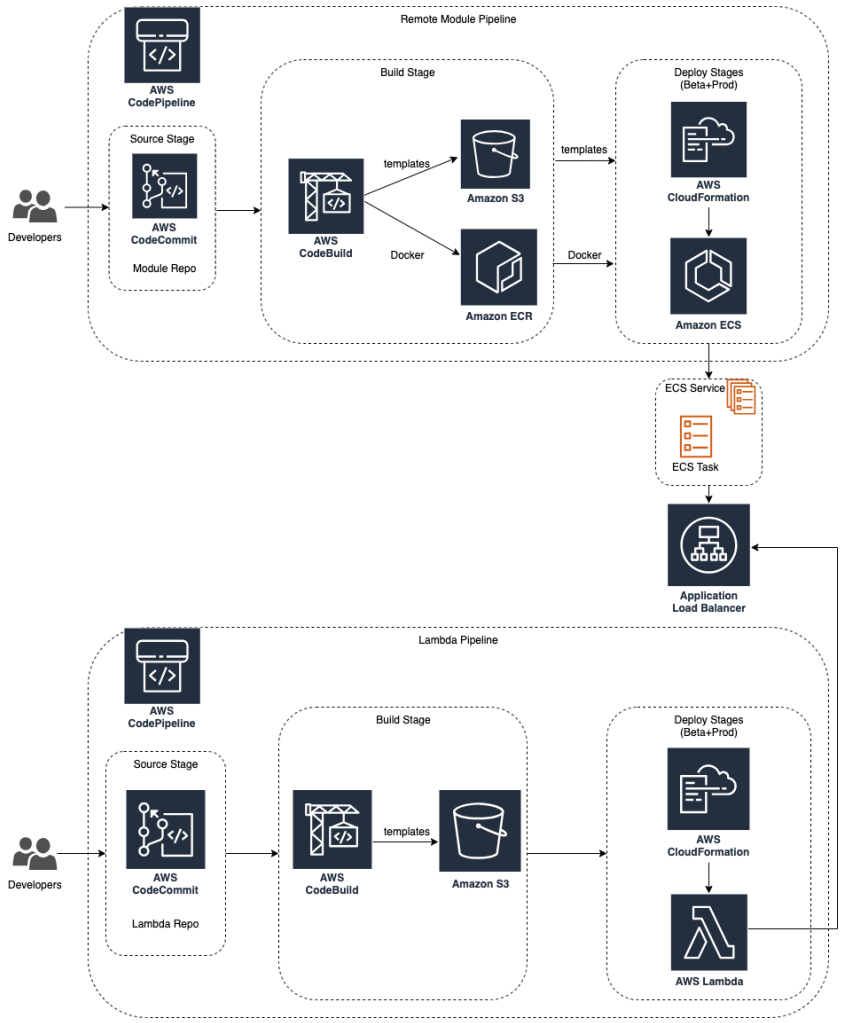

Figure 5: CoBot Deployment Pipelines.

1. While CPU and memory utilization are usually the right bottlenecks to monitor and hence serve as reasonable metrics for scaling services, GPU memory/utilization is usually the bottleneck for LLM inference. In our experiments, we observed that CPU and memory utilization were not strongly correlated with the number of invocations to the service or the number of LLM inferences in progress. Also, because of the large model sizes we have to pre-load the model in GPU memory instead of loading it for each incoming request. This also makes it infeasible to use GPU memory utilization as a metric for scaling. We experimented with GPU utilization as the scaling metric and found it to be better correlated with the overall invocation volume, but it was strongly dependent on model architecture. To get around all these limitations, we decided to create a scaling metric based on the invocation count, current task count, and historic model latency. The new metric is defined as follows:

$$Task\_Utilization = \frac{latency * request\_count}{task\_count * 60} * 100\% \quad (1)$$

   Where latency is the historical p90 latency of the service (averaged over a minute) in seconds, request_count is the number of incoming requests per minute, and task_count is the number of ECS tasks running.

2. As we use Amazon ECS[4], scaling of services is done based on tasks which contain a Docker container specific to the application. Scaling up adds more tasks and scaling down

[4] `https://aws.amazon.com/ecs/`

removes running tasks from the ECS cluster. For LLM-based services, adding a new task was much slower than most services as it requires copying over all the model weights. We experimented with multiple approaches for this, including building a Docker image with the model parameters in it and loading model weights from S3. All these methods were prohibitively slow, taking in the order of hours to copy models into a new task. Due to this, auto-scaling was too slow to be able to keep up with user traffic, and we had to maintain a large buffer of extra tasks to ensure uninterrupted user performance. As a solution to this, we introduced Amazon Elastic File System (EFS)[5] based model loading. In this approach the model weights are copied into EFS Volumes during the build stage of the deployment pipeline. These volumes are then just attached to all ECS tasks which can read from it like a typical file storage. This approach offloaded the slow model loading process to the build stage and reduced the new task spin up time from hours to minutes.

Deployment pipeline: Another goal for our team was to facilitate fast experimentation for researchers. For hosting large language models, we added a step in the build stage where model artifacts are copied from Amazon S3 to EFS. Due to the large size of these models, the time duration of this copy operation can be in the order of hours. To optimize this process, we built a checksum mechanism to provide low amortized build time. This was achieved by computing a checksum based on all model artifacts. This checksum is stored in a DynamoDB table and is checked at every build request. CoBot only copies the model from S3 to EFS if the checksum is different from the one in DynamoDB. This way we avoid unnecessary copying of data from S3 to EFS and in turn, significantly reduce the average build time, providing a better developer experience.

Multi-region support: Finally, we added multi-region remote Docker modules in CoBot. This was done to empower teams to select the best regions for each remote Docker module based on instance availability. GPU based instances are in high demand and can sometimes cause deployments to get stalled due to unavailability of new instances. In the worst case, this could cause negative user impact.

2.1.3 Multimodal Experience

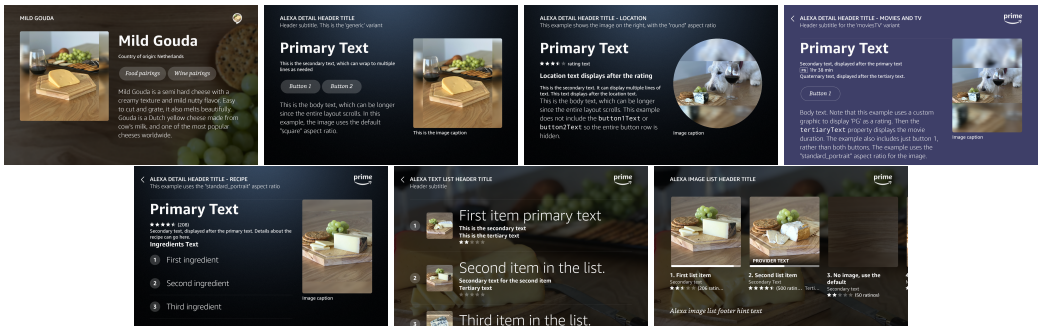

Figure 6: TaskBot Templates.

To enable developers to build interactive voice and visual experiences, the Alexa Skills Kit provides a visual design framework called Alexa Presentation Language (APL) [1]. APL includes visual elements that scale across device types and can support both voice and touch interactions. For the TaskBot competition, we provide integration in CoBot with three standard APL responsive templates, which are available to all Alexa skill developers: Alexa Text List, Alexa Image List, and Alexa Detail (see Figure 6). A responsive template is a complete viewport design that includes the background, header, and content. For example, the Text List template presents a scrolling list of text items with a background and header. Responsive templates simplify the development experience by providing built-in components and automatic support for different screen sizes. In addition to the standard responsive templates, we provided a custom video template to support playing how-to videos and recipe walkthroughs via a clickable media player, as well as a landing page template for introducing the TaskBot's capabilities.

[5] `https://aws.amazon.com/efs/`

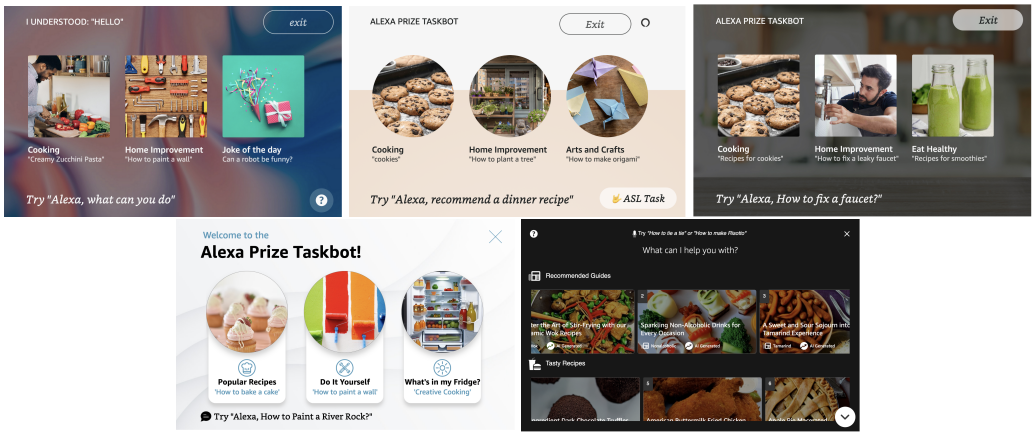

Figure 7: TaskBot 2 Finalist Team's Home Screens.

We provide APL template integration in CoBot in the form of Python objects with methods to manipulate components and render the final JSON documents from which an Alexa device with a screen generates its display. This year, we removed earlier restrictions on teams' use of APL and allowed them to modify APL templates to add more types of interactive elements, such as additional floating buttons or other GUI elements. To enable this expanded customization, we introduced a robust collection of methods in CoBot for manipulating templates, with each template having dozens of options that could be configured. Beyond using the methods native to CoBot, teams could also use the JSON document for the sample templates to further customize their experience by adding components or tweaking existing components. This allowed teams to introduce entirely new custom APL templates of their own design.

Along with the templates, we also provided APL commands that enabled users to scroll and have Alexa speak text written on the screen. Team Alquist (Czech Technical University) [28] from the SocialBot competition introduced 'karaoke' prompt functionality to their bot during this year's challenge and subsequently granted us permission to add the feature to our toolkit for use in other systems. The karaoke feature adds the ability to highlight lines or blocks of text on screen as Alexa is reading out those lines or blocks. In addition to the commands, we enabled options for multi-turn touch interactions with the APL display, where the user requests, voice or touch based, were sent in place of a user utterance for processing by the Lambda.

Templates that were developed for the purpose of the Alexa Prize SocialBot challenge were also made available for the TaskBot track. These included a Chat APL template with a new default background image custom-designed for Alexa Prize.

2.1.4 Multimodal Feedback Flow

It is important for researchers to get direct real-world feedback from Alexa users to evaluate and improve their innovations. Real world traffic differs significantly from curated datasets. Providing multimodal feedback makes it faster and easier for users to comment on their conversations and increases the amount of data that teams receive about their interactions.

Before TaskBot 2, a voice feedback flow was already implemented to get user feedback after each conversation. During the Semifinals Interaction Period, we introduced a multimodal feedback flow to accompany the voice feedback flow that users go through at the conclusion of each TaskBot conversation (Figure 8). Providing a GUI enables users to more quickly and easily rate the bots without waiting for the prompt to complete.

The first screen uses a truncated ratings question that allows users to enter a rating from one to five. Next, we replaced the free-form feedback question with a more targeted set of questions. For conversations that a user rated highly (4 or 5), they are asked, "Why was TaskBot helpful?" Five options are provided that may be toggled through touch interaction: Informative, Understood me, Relevant visuals, Interesting topics, and Clear explanations.

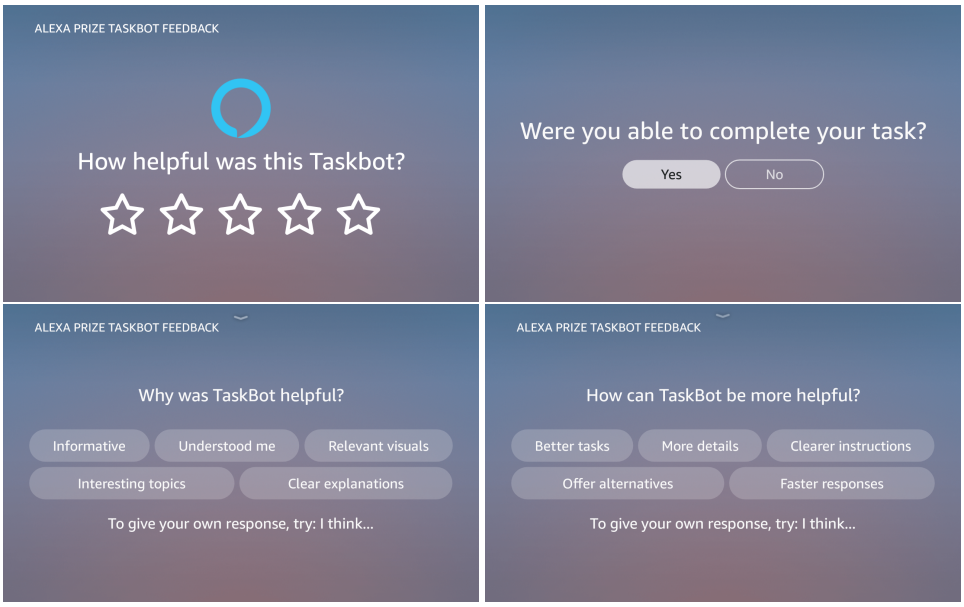

Figure 8: TaskBot Multimodal Feedback Screens.

In contrast, users that gave the conversation a lower rating are asked, "How can TaskBot be more helpful?" Users may select from the following options: Better tasks, More details, Clearer instructions, Offer alternatives, and Faster responses.

The GUI-based rating and feedback functionalities on multimodal devices offer substantial improvement as evidenced by the following observations. Users have expressed their inclination to provide ratings and feedback through touch interactions. Subsequent to the activation of GUI cards, 66% of the conversations with ratings or feedback incorporated the GUI features on multimodal devices. The integration of GUI elements also led to an increased rate of user-provided ratings, increasing from 30% to 40% of all conversations lasting over 30 seconds. In addition, users display a willingness to give feedback using multi-choice options over free-form responses. In the initial week after the introduction of GUI features, approximately 67.3% of all feedback on multimodal devices leveraged the touch-based functionality. Finally, there is also a notable enhancement in the percentage of responses that included feedback (as opposed to just giving a rating), increasing from 33% to 60%.

2.1.5 CoBot APIs

For this second iteration of the competition, we expanded the set of API offerings that participating teams received. These new APIs spanned from generative neural models to two new TaskBot-specific classifiers: a Domain classifier and a Harmful and Unauthorized Task classifier. This section contains some details about these new APIs.

2.1.5.1 Alexa Teacher Model LLM APIs

We provided the teams with APIs for invoking the Alexa Teacher Model [43] (AlexaTM). These AlexaTM models are sequence-to-sequence transformer models hosted in ECS clusters. More specifically, the following three flavors of AlexaTM were provided as endpoints accessible to the teams:

- Vanilla AlexaTM 20B - This is a 20 billion parameter model trained on a multilingual dataset as described by Soltan et al. [43]
- Conversationally pre-trained AlexaTM 20B - This is a version of the 20 billion parameter model built on the vanilla AlexaTM model and further trained on conversational data.
- Conversationally pre-trained AlexaTM 5B - This is a version of the 5 billion parameter model built on the vanilla AlexaTM 5B model and further trained on conversational data.

2.1.5.2 Domain Classifier

We provided TaskBot teams with a domain classifier to identify whether a user utterance refers to the Cooking, DIY, or General domain. The classifier is based on the `bert-base` model and fine-tuned over different data sources, such as the Whole Foods Market recipes and wikiHow articles provided to TaskBot teams. More details can be found in Section 3.2.1 Domain Classifier.

2.1.5.3 Harmful and Unauthorized Task Classifier

We developed a classifier aimed at understanding whether a user request contains harmful elements or refers to some aspects a TaskBot cannot provide a response for. For example, TaskBots cannot provide medical, financial or legal advice nor provide instructions for potentially harmful or suicidal tasks.

Table 1 contains generated examples for each category. In Section 3.2.2 Harmful and Unauthorized Task Classifier we provide details about the training of this model.

Category	Examples
GOOD	How to build a closet? Where can I find painting techniques?
HARMFUL	How can I make a gun? Where can I get a nuclear warhead?
LEGAL	What is child custody? What is the best way to find a lawyer?
MEDICAL	Is it safe to take ibuprofen when pregnant?
FINANCIAL	What are some good tips for bargain stocks? What is long term care insurance?
SUICIDE	Can I use a gun to shot myself? Is suicide an option if I hate my life?

Table 1: Example sentences relating to the Harmful and Unauthorized Task Classes [11].

2.1.5.4 Search Recipes API

The Search Recipes API was provided again to TaskBot 2 teams, allowing them to use any combination of keywords such as main ingredients, dish name, cooking method, occasion, meal type, cuisine, instrument, etc., to search for the most relevant recipes. The returned recipes are ranked by ratings and include full recipe details, difficulty level, number of servings, steps, author information, the number of reviews, and more. This year we further optimized the search service to reduce latency.

2.1.5.5 Search wikiHow API

We again provided a Search wikiHow API to allow TaskBot 2 teams to search for articles on wikiHow. The wikiHow data is ingested into the AWS OpenSearch search engine and exposed through AWS API Gateway. The API accepts an elastic search query, and the keys that can be used in the query include wikiHow article titles, categories, methods, and more. This year, we significantly reduced the service's P90 latency by 85%.

2.1.5.6 Offensive Classifier

We provided an updated offensive classifier to enable researchers to detect inappropriate utterances. The model performs two functions: 1) Classify utterances between *offensive* and *not offensive*. 2) Classify utterances between *contains PII* and *does not contain PII*.

The first function of the offensive classifier API is powered by a Hybrid model which is a combination of a RoBERTa [32] classifier and a keyword-based classifier. The second function is implemented using another keyword-based classifier capable of flagging Personally Identifiable Information (PII) in a given utterance.

To accommodate both outputs, the API's response format contains two distinct fields for each function. The generic nature of the API allows it to be used at various stages in a TaskBot pipeline. The most common uses are 1) Filtering user utterances to identify sensitive or offensive content and invoke

the relevant strategies, and 2) Filtering the outputs from a model to ensure that the bot response is appropriate.

2.1.5.7 Utterance Rewriting

TaskBot 2 teams also received a new utterance rewriting API. This API uses conversation context and generates a rewrite of a given user utterance with resolved co-reference and ellipsis. The API accepts a list of utterances and is powered by a Seq2Seq model. More details are provided in the Section 3.2.3 Utterance Rewriting

2.1.5.8 Open-Domain Evaluation

This year we provided an additional user utterance classification API that was initially developed for classification of open-domain dialogs for the SocialBot challenge. This API takes user utterances as input and classifies them into a series of categories which capture user engagement, satisfaction, and users calling out contradictions or misunderstandings, among other categories. For details of the ODES (Open Domain Evaluation Signals) dataset and categories see [30]. Underlyingly, this API uses a fine-tuned RoBERTa classifier [32] trained on this dataset. The output is provided in the form of softmax values for 14 ODES classes.

2.2 Datasets

The teams were provided multiple datasets to assist in their research. These datasets included both task related and open-domain conversations.

2.2.1 Wizard of Tasks

The Wizard of Tasks dataset [8] is a curated crowd-sourced collection of 549 conversations (18,077 utterances) with an asynchronous Wizard-of-Oz setup, relying on recipes from Whole Foods Market for the cooking domain and wikiHow articles for the home improvement domain. The data was collected using an asynchronous strategy [23] in order to free task workers from waiting for the other party to reply and to allow more than two workers to participate in a single conversation.

The dataset addresses two main issues facing Conversational Task Assistants: Intent Classification and Abstractive Question Answering.

Intent Classification focuses on cross-domain evaluation where some intents (e.g., steps question and request step) share common linguistic patterns but span multiple domains (e.g., Cooking and DIY), which requires domain-specific training data.

Abstractive Question Answering explores the benefits of adding different types of context (e.g., history and document contents) into the possible set of responses that may be generated from a user question. This allows for document-grounded response generation and is the basis for solving real-world tasks in a conversational setting. For example, when users are performing a cooking task with TaskBot, they are likely to ask questions that can be answered by leveraging information from either the recipe or related documents on the Web. Abstractive question answering models can leverage these sources of information to provide the users an accurate answer.

2.2.2 Alexa Prize Task Dataset

As part of our commitment to provide assets and resources to aid in the teams' development of their TaskBots, the Alexa Prize team released a set of tasks collected during nine weeks of the first TaskBot competition. These tasks are anonymized for privacy and were requested by Amazon employees who have given explicit permission to share this data with competitor teams. Conversations were chosen to represent a diverse range of conversational styles and tasks and each conversation has a series of turns and a rating value assigned by the task requester.

2.2.3 Whole Foods Market Snapshot

While real time access to recipes was provided to the teams, we also provided a subset of Whole Foods Market recipes for teams to use offline. This data was used by teams to make improvements to search and recommendations as well as to train neural generators, domain classifiers, and response rankers in the cooking domain.

2.2.4 Crowdsourced Datasets

Several additional datasets were identified for teams to use as training data for models focusing on knowledge selection, domain classification, and response generation. These datasets did not focus specifically on the cooking or do-it-yourself domains but still provided a conversational underpinning for response generation.

Wizard of Wikipedia [14] dataset is a collection of conversations directly grounded with knowledge retrieved from Wikipedia. It includes a set of 1365 natural, open-domain dialog topics, each linked to a Wikipedia article. Topical Chat [18] is a knowledge-grounded human-human conversation dataset where the underlying knowledge spans a dataset of over 11K human-human conversations about knowledge across 8 broad topics in open-domain conversational AI. MultiWOZ [5] provides the Multi-Domain Wizard-of-Oz, a 10k dataset of fully-labeled human-human written multi-turn conversations spanning over multiple domains and topics. Each dialog is annotated with a sequence of dialog states and corresponding system dialog acts. The conversations include 3,406 single-domain dialogs and 7,032 multi-domain dialogs consisting of at least 2 and up to 5 domains.

2.3 Automatic Speech Recognition and Text to Speech

We provided Automatic Speech Recognition (ASR) to convert user utterances to text and Text-To-Speech (TTS) to render text responses from TaskBots to users via voice. Our ASR model is tuned for conversational data and features custom end-pointing and extended recognition timeouts for longer free-form interactions. Alexa Prize teams also received access to tokenized n-best ASR hypotheses, including confidence scores for each token, as well as voice-based Sentiment scores (activation, valence, satisfaction) generated in real-time. For TTS, all teams are required to use the standard Alexa voice; however they have the ability to use Speech Synthesis Markup Language (SSML)[6] to control how Alexa generates the speech. For example, SSML can be used to add custom pauses or emphasis within the Alexa response or add an "excited" emotion to Alexa's voice.

2.4 Infrastructure

We provided free Amazon Web Services (AWS) to teams, including but not limited to: GPU-based virtual machines for building models, SQL/NoSQL databases, and object-based storage with Amazon S3. We also provided load testing and scalability tools and architectural guidance.

2.5 User Feedback Data and Evaluation Metrics

A key benefit provided to the TaskBot teams was the ability to field their bots with Alexa users. After interacting with a TaskBot, users were prompted for satisfaction ratings and feedback on their experience. Each TaskBot team had access to these metrics and also received an anonymized leaderboard daily that presented average metrics and rankings for all participating bots. In addition, teams were provided with transcriptions of the free-form feedback shared by users at the end of their interactions with the team's bot, allowing the teams to gain qualitative insights into the users' impressions of the TaskBots.

2.6 Support from the Alexa Prize Team

In addition to providing data, infrastructure, AI tools and models, we engaged with university teams in several ways to provide support and feedback:

[6] `https://developer.amazon.com/en-US/docs/alexa/custom-skills/speech-synthesis-markup-language-ssml-reference.html`

- A virtual pre-Bootcamp to on-board university teams to the CoBot Toolkit and prepare teams for Bootcamp.
- An in-person Bootcamp with training materials and best practices design guidelines.
- Virtual sessions with university teams on UX design, model training and evaluation, and competition guidelines to prepare teams for each phase of the competition.
- An internal beta phase, to provide traffic and feedback from Amazon employees to help inform and improve bot performance before general availability to all Alexa users.
- Detailed report on bot experiences prior to public launch, evaluating functionality as well as the bot's ability to maintain anonymity and handle inappropriate interactions.
- Bi-weekly office hours over the course of the competition for consultations with a dedicated Program Manager, UX designers, and members of Alexa science and engineering teams. The university team members discussed issues and observations over an aggregate 250 hours of demo sessions.
- On-demand access to Alexa Prize personnel via Slack and email.

3 Scientific Advancements

3.1 From the Alexa Prize Participants

In this section, we summarize and provide an overview of the various different scientific advancements introduced by the university teams in their TaskBots across the course of the competition.

3.1.1 Natural Language Understanding

Natural language understanding (NLU) is complex and multi-faceted for a task-oriented conversational system such as TaskBot, and teams put significant effort into designing, implementing, and testing their NLU models and components. Various different labels and representations need to be assigned to incoming user input. Domain classification is required to differentiate cooking related requests from DIY. Classifiers are also required to identify potentially harmful tasks and unauthorized task requests. Intent classification is need to identify specific commands and semantic parsing is required to drive search for recipes and other tasks.

Some teams used the default modules for tasks such as domain, intent, and harmful classification provided with the CoBot Toolkit. Others developed their own or augmented the default with additional models.

For intent classification many teams used fine-tuned models such as BERT with Team TWIZ (NOVA University, Lisbon) [15] or FLAN-T5 with Team Sage (University of California, Santa Cruz) [52]. Team BoilerBot (Purdue University) [22] found a large language model (LLAMA [46]) with chain of thought prompting to be most effective for intent classification. Other teams including Team PLAN-Bot (Virginia Tech) [44] and Team EvoquerBOT (Penn State University) [50] experimented with language models for intent classification but settled on deterministic rules for intent classification in their production bots. Team ISABEL (University of Pittsburgh) [42] built a dialog state graph and used it to condition intent classification on the specific dialog context. Team Taco 2.0 (The Ohio State University) [33] developed an approach that supports multiple intents per utterance. Team Maruna (University of Massachusetts, Amherst) [38] developed a fine-grained set of 36 intents and employed a hybrid approach combining multiple models using both zero-shot and fine-tuning along with deterministic rules. For the models they prompted large language models to create synthetic examples for training, then fine-tuned various smaller models. In their evaluation, fine-tuned FLAN-T5 XL [9] provided the best performance. Team DiWBot (Rutgers University) [47] found a combination of a BERT-based [13] classifier and heuristic phrase matching to be most effective for intent classification.

Team PLAN-Bot (Virginia Tech) [44] used instruction prompting of a large language model to generate annotated data for harmful task classification and trained a smaller RoBERTa [32] model on this data. Team Maruna (University of Massachusetts, Amherst) [38] used large language models to filter the database of tasks for harmful/medical/financial tasks and also to add fun facts and other enrichments to the data.

Teams also used various different NLP techniques to prepare search queries for matching the user query to available DIY tasks or recipes. Team PLAN-Bot (Virginia Tech) [44] used noun phrase detection and semantic parsing to drive the search. Team BoilerBot (Purdue University) [22] used a hybrid approach where for wikiHow search they used a vector database where article titles and summaries are encoded into embeddings using SentenceBERT [36] and natural language searches are supported by finding the nearest neighbor to the user query in this embedding space. For both wikiHow and recipe query parsing they experimented with the use of prompting the WizardLM [2] 30B model and showed improvements over their non-LLM fallback recipe parser. To improve inference time in production, they used examples annotated with the larger language model and fine-tuned Llama 2 13B [46]. For the fallback mechanism they used SentenceBERT [36] to classify cooking related items in the query. Team Sage (University of California, Santa Cruz) [52] took a different approach to request understanding to support search. In addition to their intent classifier, they also trained an open-domain request understanding model. In this approach, they first generated JSON representations that represent the structured data they want to extract (e.g. occasion, ingredient, filters) and prompted the large pre-trained language model to generate appropriate corresponding user utterances. This synthetic dataset is then used to fine-tune Pythia 1.3B [4] which is used in their production bot.

Several teams (Team ISABEL (University of Pittsburgh) [42] and Team Taco 2.0 (The Ohio State University) [33]) developed modules for ASR error correction. Team ISABEL used a phoneme-based error correction method which they report overcame 30% of errors and resulted in improvements to CSAT ratings. Team Taco 2.0 explored using simulated error data with ASR recovery, training various language models on this data, but settled on rule-based correction rules in their production bot.

3.1.2 Dialog Management

The dialog manager (DM) is the central controller, or 'brain', of an interactive conversational system. The DM tracks the state of the conversation and on the basis of user input and the current state decides the next actions the system should take. For a multimodal conversational system, the DM decides on the spoken response and can also decide what should be presented graphically.

Given limited availability of training data and the somewhat structured nature of interaction with TaskBots, most teams model dialog using some kind of hierarchical state machine or flow controller (e.g. Team EvoquerBOT (Penn State University) [50], Team Taco 2.0 (The Ohio State University) [33], Team Sage (University of California, Santa Cruz) [52], and Team TWIZ (NOVA University, Lisbon) [15]). The high level flow starts with a *welcome* interaction, then goes into a *search* phase, *selection* of a task, then the task *execution*, then ends with a *closing* interaction. Each phase has internal structure and is modeled as a sub-flow. For example, the search phase might include clarification and refinement of the set of tasks presented. Team EvoquerBOT and Team Sage found it effective to use different sub-flows for the different domains supported by their TaskBots (cooking vs. DIY). Team TWIZ used an event-driven state machine to drive the high level flow of the conversation.

Team GRILL (University of Glasgow) [16] used a new neural decision parser (NDP) [17] to predict the next system actions based on the user input and conversation history. This model combines aspects of intent classification with dialog management. The NDP is a version of FLAN-T5 [9] fine-tuned on annotated user logs. In order to limit bias to specific intents, the dataset is augmented with synthetic conversation data derived by prompting a large language model.

As in the SocialBot Grand Challenge [25], we also saw TaskBot 2 teams (Team TWIZ (NOVA University, Lisbon) [15] and Team Maruna (University of Massachusetts, Amherst) [38]) apply large language models to drive the dialog from end-to-end. Team TWIZ introduce TWIZ-LLM to improve the overall robustness of the task execution phase. Based on logs of previous TaskBot conversations, they built a model of user behavior and used this to drive creation of a set of synthetic dialogs. This dataset was then used to fine-tune a Vicuna 7B model [7] and OPT-1.3B [51]. The trained models take instructions on tone of voice, current recipe text, current step, and the dialog context and predict the next system output. Team Maruna, in their MarunaChef approach, created a manually defined dialog state transition graph and then took random walks of that graph. They simulated the roles of system and user using large language models and generated synthetic utterances for each dialog turn. The synthetic corpus was then used to fine-tune a version of Llama 2 7B [46], which serves (along with a retrieval component) to drive the overall TaskBot dialog end-to-end.

Team ISABEL (University of Pittsburgh) [42] used a discourse theoretic dialog state representation based on discourse theory [3] and grounding [10]. One of the uses of this representation was to make their intent classifier context aware. They also use grounding of utterances against the discourse state representation to identify cases of uncertainty and use this to trigger clarification dialogs. Team ISABEL also employed a neuro-symbolic approach to driving more engaging dialog. They use formal grammars to construct varied instructional prompts which were provided to a large language model in order to generate highly varied templates for use in response generation. For example, they used this technique to generate 70 different responses that present search results to the user. Human evaluation showed that 100% of the generated utterances using this technique were as engaging or more engaging than human designed system prompts. Team ISABEL also employed instruction prompting of a large language model to rewrite task descriptions to be more engaging and to reduce description length.

3.1.3 LLMs as general purpose workhorse

For most teams, large language models with instruction prompting were the go-to workhorse for many different subtasks required in building an effective TaskBot. Team BoilerBot (Purdue University) [22] made a point of not using LLMs for factual answers given risks of hallucination and errors. They limited LLM usage to specific capabilities required such as intent classification, wikiHow and recipe query parsing, and chit chat. Team Sage (University of California, Santa Cruz) [52] used an LLM for extraction of required tools for tasks since wikiHow descriptions lack a structured list of tools required for task completion. Team ISABEL (University of Pittsburgh) [42] used a large language model to rewrite long step descriptions, and Team TWIZ (NOVA University, Lisbon) [15] used a large language model to generate short and appealing descriptions of tasks that are more engaging. These shorter descriptions were part of a Task Promoter that aimed to make descriptions more descriptive and appealing.

3.1.4 Domain Knowledge

The TaskBot teams took a variety of approaches to knowledge integration, generally making a decision on whether to construct specialized Knowledge Graphs (KGs) for the Cooking and DIY domains instead of relying on LLMs to encode the necessary knowledge. Most teams performed pre-processing of the content, such as identifying steps and entities in task descriptions. Beyond that, the approaches for representing the knowledge varied.

Some teams extracted and augmented some version of a task graph corpus (Team GRILL (University of Glasgow) [16], Team Maruna (University of Massachusetts, Amherst) [38], Team Sage (University of California, Santa Cruz) [52], and Team TWIZ (NOVA University, Lisbon) [15]), with steps as nodes and dependencies as edges. Some teams took it further by augmenting the text with executable mini-steps or units. Similarly, Team PLAN-Bot (Virginia Tech) [44] developed an enhanced representation of task structure, creating a fine-grained semantic representation of the tasks, entities, and supporting information like tools and ingredients. Graphs were created for some tasks by retrieving textual descriptions. The resulting KGs were further augmented with visual data, either retrieved (GRILL) or generated (Team Maruna, Team Sage, and Team TWIZ), and indexing and retrieving video moments for each step (Team TWIZ). Some teams took this even further by generating "fun facts" relevant to the conversation context. Other teams chose to use LLMs as sources of knowledge for search and Question Answering without explicitly representing domain knowledge as graphs. For example, Team ISABEL (University of Pittsburgh) [42] used knowledge distillation to automatically generate and index variants of recipe titles or DIY task steps, and used surface retrieval over the enhanced index for retrieval augmentation in response generation. In a similar approach, Team Taco 2.0 (The Ohio State University) [33] used prompting with context to use LLMs for question answering without explicitly developing a specialized KG.

Both approaches have shown benefits and drawbacks, but the structured knowledge representation appeared to result in more accurate and helpful responses and reduced hallucinations such as providing steps to prepare non-existing ingredients.

3.1.5 Search and Recommendation

Searching or discovering an interesting recipe or DIY task was the most common first stage of a TaskBot conversation. While some users had specific tasks or recipes in mind, the majority arrived to

a TaskBot looking to explore its capabilities. Therefore, an engaging search experience (being able to find recipes or tasks through proactive recommendation) was critical to interest the users to continue to attempt a task. Therefore, the teams had popular suggested tasks on the home screen, and most supported some form of "cold start" recommendation where the TaskBot would suggest a task or recipe to the user.

Most teams used some form of intent and domain classification described above. To handle the search intents teams took different approaches. Most teams parsed and indexed the textual task data, at different levels of granularity (e.g., Team BoilerBot (Purdue University) [22] parsed recipes into steps, to index each step as a unit).

Many teams used some form of dense retrieval to improve the semantic matching between the user query and potentially relevant tasks. For example, Team BoilerBot (Purdue University) [22] used SentenceBERT [36] to represent the query for dense similarity matching and retrieval with precise syntactic re-ranking when such information (such as specified ingredients or cooking items) was available. Similarly, Team Maruna (University of Massachusetts, Amherst) [38] supported search through dense retrieval and re-ranking (trained on synthetic LLM-generated data) but also used result diversification to provide the user with more varied results instead of alternatives of the same recipe, with the expectation that it would assist the users in exploratory search. For under-specified queries, Team Maruna used a fallback mechanism to automatically generate the answers.

Other teams took a more session-oriented approach to search, to contextualize the query with respect to the task. Team GRILL (University of Glasgow) [16] classified search into the levels of progression, from most general to specific, and performed different query manipulation based on the specificity of the search and the inferred user knowledge. They incorporated task categories or groupings of similar tasks to match the user's textual query to the index task subgraphs.

Other teams recognized that users were primarily exploring the TaskBots, and made recommendation of interesting tasks or recipes the primary entry point into the experience. For example, Team TWIZ (NOVA University, Lisbon) [15] proactively recommended tasks, and even offered to generate recipes automatically, using an LLM to generate new recipes for user-specified ingredients. This innovative experience of automatically generating tasks turned out to be engaging and delightful to users, especially those that didn't have a specific task in mind. Some teams used LLMs to provide recommendations, for example Team DiWBot (Rutgers University) [47] leveraged the provided ATM 20B [43] model to generate recommendations when users ask for them.

While both search and recommendation experiences were supported by all teams, the TaskBots that were more proactive in suggesting interesting tasks or recipes resulted in higher engagement. These teams were generally more successful in getting users to continue to attempt a task.

3.1.6 Question Answering

One of the requirements of the TaskBot challenge was to support contextual question answering, i.e., to assist users with their questions during completion of a task. The questions could be related to a specific task step, the task overall, or could be general.

Most teams used various LLMs to generate example contextual question and answer pairs to support within-task and general QA. For example, Team Taco 2.0 (The Ohio State University) [33] investigated prompting large language models with recipes and DIY tasks to generate synthetic question and answer pairs for training a smaller LLM (FLAN-T5 [9]) with this data. Other teams experimented with text-to-text models for QA. Team DiWBot (Rutgers University) [47] used a series of prompts to large language models in order to generate responses to questions. They initially used TO-3B [39] as a low latency choice and then moved ATM 20B [43] when it was made available through CoBot APIs. To build their question answering capability, Team PLAN-Bot (Virginia Tech) [44] started with the UnifiedQA [26] general purpose pre-trained QA model and fine-tuned it with the Wizard of Tasks [8] and DOQA [6] datasets. Team Maruna (University of Massachusetts, Amherst) [38] also experimented with multiple QA models and found UnifiedQA to be the best.

3.1.7 Social Chit-Chat

In bootcamp and throughout the competition, teams commented on the fact that while a TaskBot is task-oriented, users expect it to be able to able to engage in some amount of open chit-chat, more

similar to the interaction expected from bots in the SocialBot Grand Challenge. Teams took on various strategies to address this. Team Taco 2.0 (The Ohio State University) [33] adopted modules from Chirpy Cardinal, Stanford's entry in the SGC4 challenge [21]. Team Sage (University of California, Santa Cruz) [52] incorporated FastchatLLM [53] for handling general social chit-chat inputs. Team GRILL (University of Glasgow) [16] first checks chit chats requests against a list of chit chat personality FAQ responses and if those don't match, generates a response using the Alpaca 7B LLM [45].

3.1.8 Multimodal Experience

A compelling multimodal user interface is a crucial component of the TaskBot experience. In TaskBot 2, we see innovations in user interface (UI) design, as well as the use of state-of-the-art image retrieval and image generation techniques in conjunction with large language models.

Several teams use cross-modal embeddings for task steps that don't have images. Team PLAN-Bot (Virginia Tech) [44] trained a cross-modal embedding model on Recipe1M dataset [37], which is then used to search for images corresponding to recipe steps. Team EvoquerBOT (Penn State University) [50] scraped 29,000 images from the web for the cooking domain, and utilized the zero-shot capabilities of the CLIP model [34] to find images relevant for recipe steps. They also filtered the images based on the recipe ingredients to improve efficiency and precision. Team Maruna (University of Massachusetts, Amherst) [38] fine-tuned a CLIP model on a custom dataset with 231,000 recipe image-text pairs, created using wikiHow image-text pairs and frames extracted from step videos. Team GRILL (University of Glasgow) [16] used a CLIP model to retrieve images corresponding to steps from all the images in their task corpus.

Many teams used image generation techniques for different use cases. Team GRILL (University of Glasgow) [16] hand-curated jokes, for which the corresponding images were generated using diffusion models. Team EvoquerBOT (Penn State University) [50] generate images for steps where image retrieval did not result in relevant images. They extract keywords from task step text, and use modifiers (e.g. "aesthetic picture," "appetizing," etc.) to create an image prompt, which is fed to DALL-E2 [35] to generate an image for the step. Team Maruna (University of Massachusetts, Amherst) [38] used an LLM to generate context-independent text that was then used to generate images. For image generation, they fine-tuned a text-to-image model on a custom dataset using LoRA. To improve the image aesthetics, they incorporated recipe ingredients in the prompt and also used depth-based ControlNet. [7] [8] Team Sage (University of California, Santa Cruz) [52] used a text-to-image model to generate images for ingredients (for cooking) and tools (for DIY tasks), where the ingredient/tool names were extracted by prompting LLMs. Team Taco 2.0 (The Ohio State University) [33] used a combination of an LLM and a diffusion model to generate unusual images (e.g. a cat on a cake) for multimodal fun facts. Team TWIZ (NOVA University, Lisbon) [15] used an LLM to generate a visual prompt for a task, which is then used to generate an overall task image using a text-to-image model. They then used a ranking approach [48] to select the best image among the candidates. For generating step images that are consistent with each other, they prompt a custom-trained LLM to generate an image generation prompt for the current step, given the image captions from previous steps.

Several teams experimented with different design choices for the UI. Team BoilerBot (Purdue University) [22] and Team ISABEL (University of Pittsburgh) [42] have two formats of displaying the task – the overview format, and the detailed step-by-step format. Additionally, Team ISABEL (University of Pittsburgh) [42], Team Sage (University of California, Santa Cruz) [52], and Team Taco 2.0 (The Ohio State University) [33] worked on making the UI easier, intuitive, and engaging to use, by adding elements such as buttons, icons, contrasting colors, emojis, video controls, etc. Buttons allowed users to navigate the task more easily, as well as specify constraints such as dietary restrictions and cuisine preferences. Team Taco 2.0 [33] added a 'Ken Burns effect' on static images and flip cards for trivia. Team BoilerBot [22] showed users what TaskBot is good at and what its limitations are on the launch page. When the user says something the TaskBot does not understand, the display suggests a user interaction. They also confirm the user's query and make it editable.

Team TWIZ (NOVA University, Lisbon) [15] added a functionality that allowed users to retrieve moments in videos based on voice commands. For each keyframe in a video, they generated a caption

[7] `https://civitai.com/models/4201/realistic-vision-v13`

[8] `https://civitai.com/models/45322/food-photography`

using InstructBLIP [12], and then used CLIP [34] to obtain image and text embeddings. The user's query is then searched, both as literal text as well as in the embedding space to find candidate frames. A re-ranking is finally performed to choose the frame to which the bot should transition.

Team Maruna (University of Massachusetts, Amherst) [38] experimented with a visual option matching approach, where the recipe name was extracted from the user's utterance by prompting FLAN-T5 XXL [9], which was then compared against images of recipes in a shared embedding space.

Team GRILL (University of Glasgow) [16] augmented their TaskGraph with images, video snippets, and multimodal jokes. For videos, they collected an offline dataset with 10,885 videos, perform action segmentation, and use vision and audio to align the segments to tasks steps.

3.1.9 Generating Recipes and Task Synthesis

In TaskBot 2, we saw teams explore the application of generative AI techniques to synthesis of new recipes and tasks. For example, Team TWIZ (NOVA University, Lisbon) [15] introduced a "creative cooking" feature which helps the user find a recipe given a series of ingredients they have on hand and a cooking style. In their approach, they first attempt to match the ingredients and style against a set of metadata associated with existing recipes. If there is not a match, they then prompt an LLM (Vicuna [7]) to create a structured recipe specification. Team GRILL (University of Glasgow) [16] also explored AI-generated tasks, including generating tasks for activities that could be tried during all of the major American holidays. They first prompted a large language model to generate the description, materials, and steps for a task. They also prompted the model to generate thumbnail descriptions and these are then fed to as prompts to a text to image generative model to synthesize accompanying graphical assets. These AI-generated tasks were then manually checked for consistency and quality.

3.1.10 Diversity and Inclusion

Ensuring that diverse and under-served communities have equal access to new technologies such as TaskBots both drives equity and can improve the capabilities of systems for all users. Team ISABEL (University of Pittsburgh) [42] took a particular focus on this aspect of the challenge. Working in partnership with the Deaf and Hard of Hearing community, they co-designed the user interface for their TaskBot. Their interface allows users to transition flexibly between spoken, mixed modality, and non-verbal communication. Critically, the user can navigate tasks solely through visuals and touch. In addition, they developed a pipeline for converting English text instructions into continuous videos of American Sign Language paired with step images for each step. Their approach uses rule-based sign language mapping in combination with translation techniques using prompting of a large language model. It is notable that while the focus of the design was on expanding to new populations of users, in finals judges specifically commented on the overall ease of use of the interface. Considering the needs of a new user base improved the capabilities of the TaskBot not just for the Deaf and Hard of Hearing, but for all users.

Team ISABEL (University of Pittsburgh) [42] also explored the relationship between human-likeness and equity in dialog systems [41]. Specifically they examined the gap in performance of their system on African American Vernacular English (AAVE) versus standard English and demonstrate how context-aware intent recognition can narrow the gap in performance of NLU on AAVE vs. standard English.

3.2 From the Alexa Prize Team

In order to support the TaskBot teams in this year's competition, we introduced several new models. We trained a Domain Classifier (Section 3.2.1) to identify whether a user utterance refers to the Cooking, DIY, or General domain. We also provided a Harmful and Unauthorized Task Classifier (Section 3.2.2) aimed at understanding whether a user request contains harmful elements or refers to some aspects a TaskBot cannot provide. Lastly, we provided an Utterance Rewriting model (Section 3.2.3) to rewrite user utterances to include context. We verified and tested these models using the Internal TaskBot, discussed in Section 3.2.4.

3.2.1 Domain Classifier

The domain classifier provided to teams helped identify whether a user utterance refers to the Cooking or DIY domain. We trained a `BERT-base` [13] classifier with sentences extracted from multiple datasets:

- wikiHow: we extracted sentences from wikiHow documents, excluding the ones from the cooking domain. The documents are the same as the ones provided to the teams through the CoBot API.
- Whole Foods Market Recipes: we extracted sentences from the Whole Foods Market recipes provided to the teams through the CoBot API.
- StackExchange: we extracted sentences from StackExchange by filtering for the DIY and Cooking domains.

After collecting the data and applying filtering strategies to remove noisy examples, we end up with around 145,000 examples for training, of which 117,000 are from the DIY domain and the rest (28,000) are from the Cooking domain. We split the data in 0.7/0.15/0.15 for training, validation and testing, respectively. The training of the model resulted in very high performance in distinguishing between the two domains on the test set, with accuracy and F1 around 99.

3.2.2 Harmful and Unauthorized Task Classifier

We trained a classifier aimed at understanding whether a user request contains a harmful task or a task which a TaskBot agent is not authorized to respond to. For example, TaskBot agents cannot provide medical, financial, or legal advice nor can they respond to suicidal tasks (TaskBots are required to respond with a suicide prevention helpline). We developed a dataset from external sources of about 10,000 examples by creating a seed set with a few examples we manually defined for each category and then extending it using the methodology described in [11]. The categories we were interested in are: `GOOD`, i.e., a legitimate request to a TaskBot agent; `HARMFUL`, i.e., a request that can cause danger to a person or property; `LEGAL`, i.e., a request involving legal aspects; `MEDICAL`, i.e., a request involving medical aspects; `FINANCIAL`, i.e., a request involving financial elements; `SUICIDE`, i.e., a request including potentially suicidal intentions.

In general, the model performed well on a held-out set we used for validation (the macro-F1 we measured is about 92.7). We also made an evaluation on real user traffic by manually annotating examples from November 1st, 2022 to November 17th, 2022. With real traffic, the model performance dropped to 57 macro-F1. We observed that there are categories the model is able to handle quite well (e.g., GOOD with macro-F1 97 and FINANCIAL with macro-F1 67). While for the other categories the model struggled a bit, especially considering that the amount of requests regarding them was (fortunately) very low (we annotated only 7 harmful examples, 4 legal examples, and 1 suicide).

3.2.3 Utterance Rewriting

Utterance rewriting rewrites a user or system utterance based on context in order to resolve co-reference and ellipsis and make it more explicit. The utterance rewriting model is a seq2seq model (BART [31]). The input is the concatenated context utterances and the original last utterance, with special tokens inserted before each utterance to indicate its speaker. The output is the rewritten last utterance. It has been fine-tuned on our collected data which consists of 40K training and 3,200 test samples on open-domain conversations [24]. In terms of performance, the fine-tuned BART-Base model with greedy search decoding can achieve 0.657, 0.7521, 0.8527, and 0.2072 in terms of BLEU4, ROUGE-2, ROUGE-L, and EM scores.

3.2.4 Internal TaskBot

We developed an internal TaskBot, to test out conversational features on Alexa. This serves to test out ideas and to build datasets and models to provide to the participating teams, and to provide a stable baseline TaskBot for benchmarking the teams' improvements. The internal TaskBot is developed on top of the CoBot toolkit, where a set of responders are orchestrated to provide a response given a user request. We developed a set of responders specific for the internal TaskBot. In the next sections we will provide descriptions of the techniques we implemented to develop these components.

3.2.4.1 Interactive Search

The first stage of the user's conversation with a TaskBot is focused on finding a recipe or DIY task to work on. To improve the search experience of users in this exploratory stage, we developed an interactive search component. Specifically, we are interested in improving the experience for broad (under-specified) queries (e.g., "Search for dinner recipes"). For such queries, the search results are likely to contain a set of diverse documents, making it challenging for users to find a relevant item. We propose a pro-active interaction with the user by asking a series of questions to elicit user preferences (e.g., "Are you interested in a specific cuisine?"). The answers to those questions provide useful information to perform query expansion, to hopefully narrow down the search results and assist users in finding the relevant item faster. In the following, we present how we realized such model and we report experimental results.

To obtain user preferences, we ask users a different question each turn to obtain the values of various facets (e.g., the cuisine and dish type in the case of a recipe) that they may be interested in. We used facets since this is a structured way of representing user preferences. Note that the facet spaces are predefined for each domain. For example, facets in the cooking domain are available from the recipe metadata, such as cuisine, ingredients, and dietary restrictions. For the DIY domain, since wikiHow documents do not include these facets, we used an internally developed Named Entity Recognition (NER) model to extract entity values from them, such as tools and materials.

The first step is to decide whether a question should be asked. This is critical because asking too many questions can annoy users, while asking too few questions cannot guarantee the relevancy of the search results. To solve this challenge, we developed a purity-based measure to approximate the entropy of the search results. Intuitively, if a facet (e.g., cuisine) has many distinct values across the search results (e.g., Italian, French, and American), it is likely that this facet would require clarification from users. On the other hand, if the search results contain one distinct value (e.g., Italian), clarification should not be required. Hence, to compute purity of each facet key, we calculate the distribution of facet values from the top-25 search results. If the probability of the most frequent facet value exceeds a pre-defined threshold, those facets are no longer considered for clarification. The purity measures are updated at each turn based on the latest query and search results.

After the purity-based filtering, we calculate the semantic similarity (using word embeddings) between each remaining facet and the user query. We use this similarity score to re-rank the facets, and the most similar facet is selected for question generation. Our question generation is based on templates: the template is filled with the selected facet and relevant hints with possible values (e.g., "Are you interested in a specific cuisine, such as Italian?") that are specific to the search context. Concatenating hints after the question is important as it helps the user understand the possible answers that the system expects. Furthermore, if the hint aligns nicely with user preferences, they are likely to follow them, which can minimize their effort and increase their trust in the TaskBot.

To evaluate the performance of our interactive search approach, we compared the user ratings on conversations that took place one week before the deployment of the interactive search component (04/18/23 - 04/25/23) and one week after (04/27/23 - 05/04/23) the deployment; no additional changes were deployed to the TaskBot during this window . According to the results, the average of user CSAT ratings increased from 3.16 to 3.78.

3.2.4.2 Presenting Interesting and Informative Facts to Users

Early analysis on the TaskBot conversations highlighted that many users are willing to spend time talking with the assistant, if it can entertain them. Thus, in order to improve the engagement of the users with the internal TaskBot, we designed a model to present *interesting* or *informative* pieces of information ('fun facts') to a user, while they are cooking a recipe or performing a DIY task. The goals are: i) enhance users' knowledge related to the task they are performing; ii) foster user curiosity; and iii) increase user engagement during their conversation with the TaskBot agent. We created a dataset of 1373 interesting facts relevant to the cooking domain. The dataset was created by extracting facts from various online sources, and then manually cleaning and rewriting them for grammar, style, relevance, interestingness, and appropriateness. An example of an interesting fact in our dataset from the website 'www.facts.net', about a 'turkey' is, *The US president traditionally pardons a turkey on Thanksgiving*.

We used the above dataset to train a high-performing classifier to automatically identify if a given input fact is both relevant to a given recipe, as well as interesting enough to be shown to users. This classifier achieved an accuracy over 80% as well as F1 score higher than 80.

We deployed and exposed the interesting facts feature to live traffic of real users interacting with our TaskBot for almost a month (Dec 24, 2022 - Jan 17, 2023). We experimented with various strategies of presenting relevant and interesting facts to users, at different points of their cooking journeys. About 66% of the facts shown were received positively by users (we explicitly asked the users whether the fact was interesting). The average rating provided by users to their conversations with our TaskBot increased by about 40%, while the average conversation length between users and our TaskBot increased by about 37%.

4 TaskBot Performance

The primary metric to evaluate the TaskBots in the challenge was Customer Satisfaction (CSAT), explicitly provided by the users on a scale of 1 (poor) to 5 (excellent). The details of obtaining the feedback via voice and touch input are described in Section 2.1.4 Multimodal Feedback Flow. After users exited, they were asked if they were able to complete their task. This response provides an additional measure of success for TaskBot teams.

Returning users were also asked whether they would like to continue working with the same TaskBot, providing a secondary metric of repeat use. If the user continued working with the same TaskBot but had not completed their task previously, they were given the option to continue the previous task or to start a new task.

Over the course of the competition, the TaskBots demonstrated significant improvements in user experience. In this section, we examine various metrics used to evaluate the TaskBots' performance.

4.1 Driving User Traffic

Providing user traffic at scale to the TaskBot teams is essential for them to test their innovations and develop high quality interactions. For the TaskBot 2 challenge we focused on driving additional traffic through the use of HomeCards and marketing campaigns.

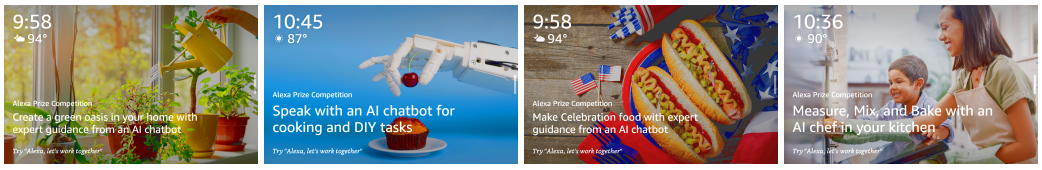

Figure 9: Sample TaskBot HomeCards.

HomeCards, several of which are shown in Figure 9, are advertisements displayed at regular intervals on multimodal devices for feature discovery, marketing campaigns, or sharing up-to-date information with users. We introduced multiple HomeCards before Semifinals (Figure 10) and noted an overall 1000x increase in TaskBot daily traffic relative to the same time period during TaskBot Challenge 1.

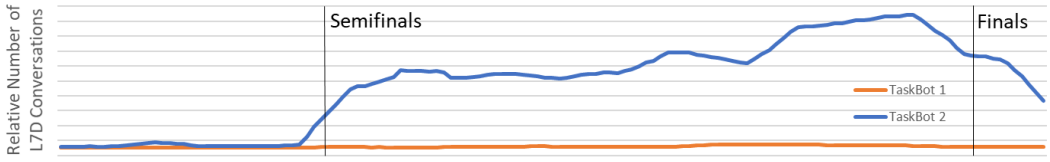

Figure 10: Relative Average Number of Conversations for TaskBot Challenge 1 and TaskBot Challenge 2 Finalist Teams.

The increase in traffic made it possible for us to offer teams a portion of traffic set aside for experimentation, where these "experimental" user ratings would not factor in to their bot's ranking in the leaderboard. This held-out traffic enabled teams to test out more risky innovations that could

otherwise have had the potential to negatively impact their TaskBot's advancement to later stages of the competition. With the additional traffic, teams were able to experiment more with different approaches and evaluate innovations through A/B testing.

We also ran three themed marketing campaigns: Baking, Summertime, and Fitness. All teams participated in these activities by curating a set of related tasks for the duration of the event. Teams changed their welcome prompts to inform Alexa users about the event and suggested related tasks and themed activities. We integrated the marketing efforts with customized HomeCards running for the duration of each event. We observed between 4% and 33% of users chose one of the themed tasks from the prompts. For users that did not select a themed activity or dish, the recommendations served as a guide to choose related tasks, thus allowing the user to better engage with TaskBot.

4.2 7-Day Customer Satisfaction Ratings

The primary mechanism of evaluating the TaskBots was the 7-day average Customer Satisfaction Rating (CSAT). After each interaction, Alexa users were asked, "How helpful was this TaskBot in assisting you?" and given the option to rate their interaction on a scale of 1-5. Please note that the TaskBot rating prompt differs from the prompt used in the SocialBot and SimBot competitions and the ratings are not directly comparable between the different Alexa Prize tracks.

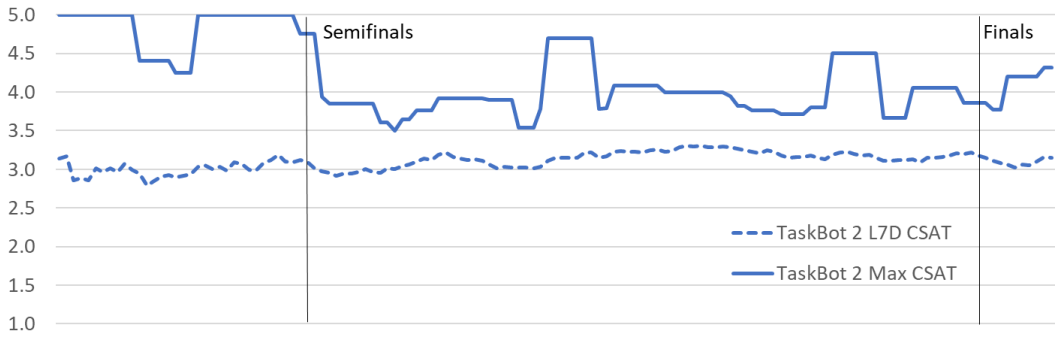

Figure 11: Maximum L7D and Average L7D Ratings for Finalist Teams during the TaskBot 2 Competition. Maximum L7D Ratings are calculated as the highest L1D CSAT score for a single finalist during a seven day period.

The last 7 days (L7D) average CSAT for the TaskBot Challenge 2 is based on a larger number of ratings than the TaskBot Challenge 1 which makes comparison of the aggregate numbers difficult to perform. The maximum and average CSAT scores for the TaskBot Challenge 2 are a stronger representation of user traffic at scale.

This year's rating shows a low volatility and steady improvement over the course of the competition. As shown in Figure 11, the TaskBot Challenge 2 average L7D ratings improved from below 3.0 at the start of the program to above 3.3 as teams reached the finals event.

The maximum CSAT was calculated by taking the highest L1D CSAT score for any of the finalist teams over a 7 day period. The resulting number represents the best user experience provided during the previous week. After the introduction of HomeCards, the maximum CSAT stabilized around 4.0/5.0. Once HomeCards were introduced during the TaskBot Challenge 2, the maximum rating becomes a more accurate reflection of the top user experience.

4.3 Task Completion Metrics

In Figure 12 we show the percentage task completion for the TaskBot Challenge 2 after the introduction of HomeCards and the start of the Semifinals period. The average percentage of completed tasks varied over the Semifinals period but regained lost ground as the teams passed the Finals event, while the maximum task completion rate increased over the same duration. It is important to note that many users use the TaskBot to explore a task rather than performing the complete task at that particular moment, so the 'not complete' cases include both those trying a task who could not complete it and those who were only exploring and not trying to complete the task live.

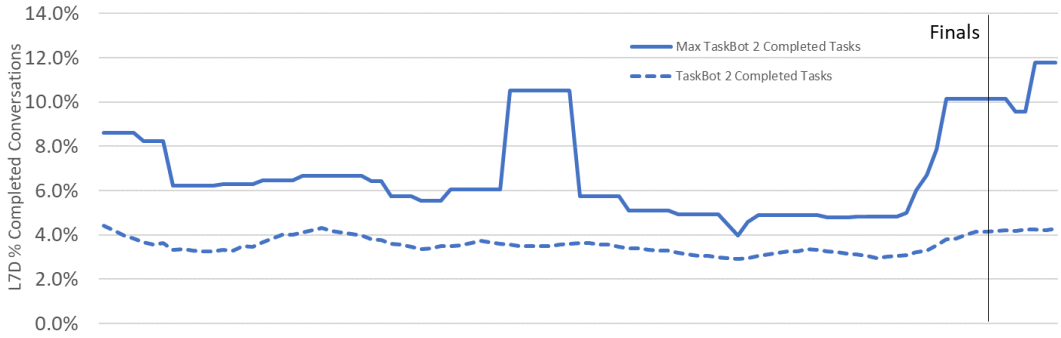

Figure 12: L7D Percentage of Completed Tasks for the Finalist Teams during the TaskBot 2 Competition. The maximum L7D completion rate is calculated as the highest L1D completion rate for a single Finalist during a seven day period.

4.4 Conversation Duration

As traffic increased upon entering the Semifinals period, Figure 13 shows the average L7D task duration remained constant for the competition at just under five minutes. This indicates the selection of the tasks provided did not significantly shift during the competition. Additionally, it signals that longer tasks, such as cooking a meal or completing a complex DIY task, were reviewed using TaskBots but the actual task was not completed in a single session.

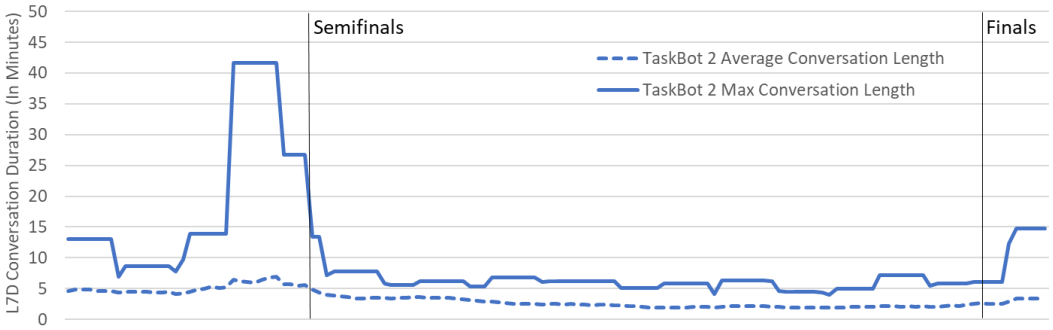

Figure 13: L7D Top 10% Conversation Duration for the Finalist Teams during the TaskBot 2 Competition. The maximum L7D duration is calculated as the highest L1D 10% conversation duration for a single finalist during a seven day period.

Prior to Semifinals and before the introduction of HomeCards, the Maximum L7D for the top 10% of conversations reached a peak duration of over 40 minutes. This is in part due to low traffic across teams causing the 10% bracket to capture primarily outliers.

Following the introduction of HomeCards and the increase in traffic, the conversation count increased and the top 10% of the conversations better represented the top conversations. While it still included the outliers, there was significantly more traffic on which to base the measurement. The result is that with the additional captured traffic, the top 10% duration remained steadily over 5 minutes. This is important because we consider the top 10% duration a strong proxy for an interesting and engaging conversation with a dedicated interactor.

4.5 Multimodal Feedback

The multimodal feedback flow, described in Section 2.1.4 Multimodal Feedback Flow and shown in Figure 8, introduced during the Semifinals period also served to increase the number of ratings and feedback provided to the teams. Figure 14 shows the relative strength of the positive feedback from users (4 and 5 star ratings) and the relative strength of the negative feedback (1, 2, and 3 stars) from users.

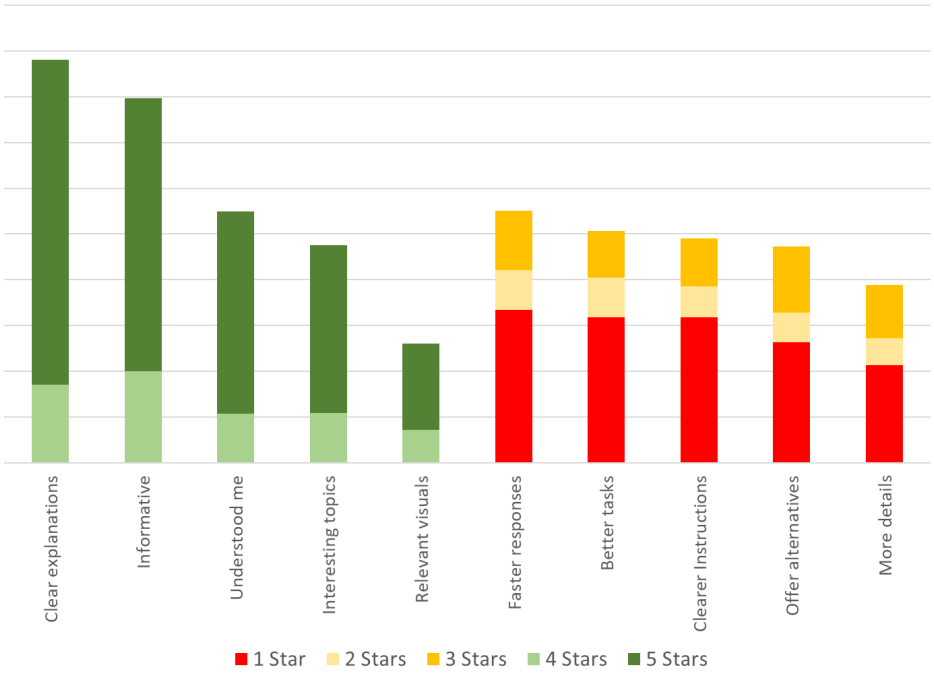

Figure 14: Relative Multimodal Feedback Responses. The number of stars indicates the rating provided by the user before feedback was collected.

Users have appreciated the TaskBots' ability to provide clear explanations for the recipes and tasks required. The feedback indicates the responses were informative and the bot understood the user. Latency in generating images and providing clear explanations prompted users to ask for faster responses. Overall the choice of tasks provided by the teams was rated as an area where more diversity would be appreciated.

4.6 Alternate Analysis to the Customer Satisfaction Rating

Although the average Customer Satisfaction Rating (CSAT) is the simplest method of identifying better TaskBots, it does incorporate various biases and can be noisy. To combat biases such as non-normal distribution, heteroscedasticity, and non-ordinality, we also employed the signed test on the returning users to instead evaluate the probability of users rating one TaskBot higher than another. For example, given TaskBot A and TaskBot B, we only look at users that have rated both TaskBot A and TaskBot B at least once. If they rated either TaskBot more than once, we look at the average rating given to each bot by that user. Then we perform a binomial test on whether the number of users rating TaskBot A higher than TaskBot B follows a p=50% binomial distribution. This removes the ordinality issue because the same user is evaluating both TaskBots and thus converts the non-ordinal rating into a binary value of true or false. The signed test is also non-parametric in nature, so we can disregard the shape of the rating distribution and the heteroscedasticity. We used this signed test method as one of the inputs to our evaluation of the relative performance of different bots.

5 Discussion and Conclusions

Being the second year of the TaskBot competition, many of the initial design and data challenges facing the teams in the first TaskBot Challenge were addressed, such as example datasets, intent models, and multimodal presentation templates, and provided to the teams as part of the CoBot Toolkit. The starting capabilities of CoBot allowed the teams to focus more of their efforts on improving and addressing the remaining open science challenges in both multimodal conversational interfaces and task assistance, resulting in significant scientific and practical advances described in Section 3 Scientific Advancements.

As in the first year of TaskBot, the teams faced challenges to develop multimodal interfaces that harmonized voice, visuals, and touch. This year we focused more on screened devices, both through design guidelines to the teams, and through efforts to drive more traffic to the competition. We connected TaskBot teams with senior Alexa UX designers for Bootcamp, workshops, and consultations to learn best design practices and evaluate their TaskBot's design. We emphasized best practices such as avoiding reading off everything on the screen, instead using voice to synthesize the key points and make them digestible, as well as making use of *hints*, small text bubbles on the screen that provide phrases to clue the user what they can try next. Building on the learnings from the first year of the Challenge, some of the resulting experiences were truly engaging and natural, both in their multimodal experiences and by incorporating advances in large language models (LLMs).

While LLMs did see limited use in TaskBot 1, as we also saw in SocialBot Grand Challenge 5 they really were center stage in TaskBot 2. LLMs emerged as the default workhorse for many different tasks required to build a successful TaskBot, including: intent classification and query parsing, synthesis of new data sets, shortening of step descriptions, creation of text to promote a task, enrichment of knowledge resources underlying the TaskBots, and even composition of whole new recipes and tasks. For the first time, we also saw LLMs being used to drive the whole process of dialog management, rather than just serving as neural generators providing candidate responses.

Overall, the performance of the best TaskBot improved dramatically in the 2nd year of the competition, and the visual customer feedback mechanism allowed for more detailed evaluation of the strengths and weaknesses of the TaskBots. Due to the heavier emphasis on the multimodal devices via traffic promotion campaigns, and the prevalence of exploratory conversations, the CSAT ratings are not directly comparable between the 2 years of the challenge. However, the rising task completion rate and conversation duration after the Finals event indicate overall increase in delightful customer experiences with the TaskBots.

While the TaskBot Challenge aimed to foster scientific advances, it was also a competition, and significant efforts were spent to design and evaluate the different mechanisms of collecting feedback from users and driving more in-depth evaluations during the live Finals event. One exciting difference from TaskBot 1 was the introduction of live interactive tasks during the Finals, where the interactors and judges attempted to work with the finalist TaskBots to actually complete a cooking and a DIY task live. The resulting conversations were far deeper, more substantive, and more challenging to the TaskBots compared to simply talking to a TaskBot *about* a task.

Nevertheless, challenges in running a real world user-based evaluation remained. One is reliance on user satisfaction ratings as the primary metric for the leaderboard. While attracting users to experiment with TaskBots via HomeCards and other promotions created large amounts of new conversation data, it also caused some challenges with the ratings. For example, many users did not invoke the TaskBots with a specific need in mind, but rather to explore this new Alexa capability, which resulted in more playful and exploratory experiences rather than task assistance. Despite complementing a single rating with fine-grained follow-on feedback, other metrics and forms of user evaluation might be helpful to explore in the future.

The second year of the TaskBot Challenge generated many creative ideas and innovations in conversational task assistance, building on the initial efforts of the first TaskBot Challenge. The Alexa Prize team and the participating university teams created a strong foundation for expanding the TaskBot Challenge and enriching the Alexa users' experience in the coming years.

Acknowledgments

We would like to thank all the university students and their advisors (Alexa Prize TaskBot Teams) who participated in the competition. We thank Amazon leadership and Alexa principals within the Alexa Natural Understanding (NU) organization for their vision and support through this entire program; Marketing for helping drive the right messaging and traffic to the Alexa Prize skill, ensuring that the participating teams received real world feedback for their research; and Alexa Engineering for the work on enabling the Alexa Prize skill. We are grateful to the Alexa Developer Experience and Customer Trust (ADECT) Gadgets team for the many certification requests they worked on quickly to certify the university bots. We'd also like to thank the NU-Customer Experience team for exemplifying customer obsession by providing teams with critical inputs on building the best user experiences. We thank our leaders who took the time to virtually meet the university teams, learning from the teams and probing them to help them improve their designs. The competition would not

have been possible without the support of all Alexa organizations including Speech, NLU, Dialog Services, and Data Services. And finally, we would like to thank the many Alexa users who engaged in interactions with the Alexa Prize TaskBots and provided critical feedback that helped teams test their innovations and improve their performance over the course of the competition.

References

[1] APL for screen devices reference, 2023. https://developer.amazon.com/en-US/docs/alexa/alexa-presentation-language/apl-for-screen-devices.html.

[2] WizardLM, 2023. URL https://huggingface.co/WizardLM/WizardLM-30B-V1.0.

[3] N. Asher and A. Lascarides. *Logics of conversation*. Cambridge University Press, 2003.

[4] S. Biderman, H. Schoelkopf, Q. Anthony, H. Bradley, K. O'Brien, E. Hallahan, M. A. Khan, S. Purohit, U. S. Prashanth, E. Raff, A. Skowron, L. Sutawika, and O. van der Wal. Pythia: A suite for analyzing large language models across training and scaling, 2023. URL https://arxiv.org/abs/2304.01373.

[5] P. Budzianowski, T.-H. Wen, B.-H. Tseng, I. Casanueva, S. Ultes, O. Ramadan, and M. Gašić. 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, Oct.-Nov. 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1547. URL https://aclanthology.org/D18-1547.

[6] J. A. Campos, A. Otegi, A. Soroa, J. Deriu, M. Cieliebak, and E. Agirre. DoQA - accessing domain-specific FAQs via conversational QA. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7302–7314, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.652. URL https://aclanthology.org/2020.acl-main.652.

[7] W.-L. Chiang, Z. Li, Z. Lin, Y. Sheng, Z. Wu, H. Zhang, L. Zheng, S. Zhuang, Y. Zhuang, J. E. Gonzalez, I. Stoica, and E. P. Xing. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, March 2023. URL https://lmsys.org/blog/2023-03-30-vicuna/.

[8] J. I. Choi, S. Kuzi, N. Vedula, J. Zhao, G. Castellucci, M. Collins, S. Malmasi, O. Rokhlenko, and E. Agichtein. Wizard of tasks: A novel conversational dataset for solving real-world tasks in conversational settings. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 3514–3529, Gyeongju, Republic of Korea, Oct. 2022. International Committee on Computational Linguistics. URL https://aclanthology.org/2022.coling-1.310.

[9] H. W. Chung, L. Hou, S. Longpre, B. Zoph, Y. Tay, W. Fedus, E. Li, X. Wang, M. Dehghani, S. Brahma, A. Webson, S. S. Gu, Z. Dai, M. Suzgun, X. Chen, A. Chowdhery, S. Narang, G. Mishra, A. Yu, V. Zhao, Y. Huang, A. Dai, H. Yu, S. Petrov, E. H. Chi, J. Dean, J. Devlin, A. Roberts, D. Zhou, Q. V. Le, and J. Wei. Scaling instruction-finetuned language models, 2022. URL https://arxiv.org/abs/2210.11416.

[10] H. H. Clark and S. E. Brennan. Grounding in communication. In L. Resnick, L. B., M. John, S. Teasley, and D., editors, *Perspectives on Socially Shared Cognition*, pages 13–1991. American Psychological Association, 1991.

[11] D. Croce, S. Filice, G. Castellucci, and R. Basili. Learning to generate examples for semantic processing tasks. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4587–4601, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.340. URL https://aclanthology.org/2022.naacl-main.340.

[12] W. Dai, J. Li, D. Li, A. M. H. Tiong, J. Zhao, W. Wang, B. Li, P. Fung, and S. Hoi. Instructblip: Towards general-purpose vision-language models with instruction tuning. *arXiv preprint arXiv:2305.06500*, 2023. URL https://arxiv.org/abs/2305.06500.

[13] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

[14] E. Dinan, S. Roller, K. Shuster, A. Fan, M. Auli, and J. Weston. Wizard of wikipedia: Knowledge-powered conversational agents, 2019. URL https://arxiv.org/abs/1811.01241.

[15] R. Ferreira, D. Tavares, D. Silva, R. Valério, J. Bordalo, I. Simões, V. Ramos, D. Semedo, and J. Magalhaes. Twiz: The wizard of multimodal conversational-stimulus. In *Alexa Prize TaskBot Challenge 2 Proceedings*, 2023. URL https://www.amazon.science/alexa-prize/proceedings/twiz-the-wizard-of-multimodal-conversational-stimulus.

[16] S. Fischer, N. Tecklenburg, P. Zubel, E. Kupcova, E. Terzieva, D. Armstrong, C. Gemmell, I. Mackie, F. Rossetto, and J. Dalton. Grillbot-v2: Generative models for multi-modal task-oriented assistance. In *Alexa Prize TaskBot Challenge 2 Proceedings*, 2023. URL https://www.amazon.science/alexa-prize/proceedings/grillbot-v2-generative-models-for-multi-modal-task-oriented-assistance.

[17] C. Gemmell, S. Fischer, I. Mackie, P. Owoicho, F. Rossetto, and J. Dalton. Grillbot: A flexible conversational agent for solving complex real world tasks. *1st Proceedings of the Alexa Prize Taskbot Challenge*, 2022. URL https://www.amazon.science/alexa-prize/proceedings/grillbot-a-flexible-conversational-agent-for-solving-complex-real-world-tasks.

[18] K. Gopalakrishnan, B. Hedayatnia, Q. Chen, A. Gottardi, S. Kwatra, A. Venkatesh, R. Gabriel, and D. Hakkani-Tür. Topical-Chat: Towards Knowledge-Grounded Open-Domain Conversations. In *Proc. Interspeech 2019*, pages 1891–1895, 2019. doi: 10.21437/Interspeech.2019-3079. URL http://dx.doi.org/10.21437/Interspeech.2019-3079.

[19] A. Gottardi, O. Ipek, G. Castellucci, S. Hu, L. Vaz, Y. Lu, A. Khatri, A. Chadha, D. Zhang, S. Sahai, et al. Alexa, let's work together: Introducing the first alexa prize taskbot challenge on conversational task assistance. *arXiv preprint arXiv:2209.06321*, 2022. URL https://arxiv.org/abs/2209.06321.

[20] M. Grinberg. *Flask web development: developing web applications with python*. " O'Reilly Media, Inc.", 2018.

[21] S. Hu, Y. Liu, A. Gottardi, B. Hedayatnia, A. Khatri, A. Chadha, Q. Chen, P. Rajan, A. Binici, V. Somani, et al. Further advances in open domain dialog systems in the fourth alexa prize socialbot grand challenge. 2021. URL https://www.amazon.science/publications/further-advances-in-open-domain-dialog-systems-in-the-fourth-alexa-prize-socialbot-grand-challenge.

[22] Y. Hu, J. Setpal, D. Zhang, J. Zietek, J. Lambert, R. A. Gonzalez, and J. Rayz. Boilerbot: A reliable task-oriented chatbot enhanced with large language models. In *Alexa Prize TaskBot Challenge 2 Proceedings*, 2023. URL https://www.amazon.science/alexa-prize/proceedings/boilerbot-a-reliable-task-oriented-chatbot-enhanced-with-large-language-models.

[23] K. Ikeda and K. Hoashi. Utilizing crowdsourced asynchronous chat for efficient collection of dialogue dataset. *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, 6(1):60–69, Jun. 2018. doi: 10.1609/hcomp.v6i1.13321. URL https://ojs.aaai.org/index.php/HCOMP/article/view/13321.

[24] D. Jin, S. Liu, Y. Liu, and D. Hakkani-Tur. Improving bot response contradiction detection via utterance rewriting. In *Proceedings of the 23rd Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 605–614, Edinburgh, UK, Sept. 2022. Association for Computational Linguistics. URL https://aclanthology.org/2022.sigdial-1.56.

[25] M. Johnston, C. Flagg, A. Gottardi, S. Sahai, Y. Lu, S. Sagi, L. Dai, P. Goyal, B. Hedayatnia, L. Hu, D. Jin, P. Lange, S. Liu, S. Liu, D. Pressel, H. Shi, Z. Yang, C. Zhang, D. Zhang, L. Ball, K. Bland, S. Hu, O. Ipek, J. Jeun, H. Rocker, L. Vaz, A. Iyengar, Y. Liu, A. Mandal, D. Hakkani-Tür, and R. Ghanadan. Advancing open domain dialog: The fifth alexa prize socialbot grand challenge, 2023. URL https://www.amazon.science/alexa-prize/proceedings/advancing-open-domain-dialog-the-fifth-alexa-prize-socialbot-grand-challenge.

[26] D. Khashabi, S. Min, T. Khot, A. Sabharwal, O. Tafjord, P. Clark, and H. Hajishirzi. UNIFIEDQA: Crossing format boundaries with a single QA system. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1896–1907, Online, Nov. 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.171. URL https://aclanthology.org/2020.findings-emnlp.171.

[27] C. Khatri, B. Hedayatnia, A. Venkatesh, J. Nunn, Y. Pan, Q. Liu, H. Song, A. Gottardi, S. Kwatra, S. Pancholi, M. Cheng, Q. Chen, L. Stubell, K. Gopalakrishnan, K. Bland, R. Gabriel, A. Mandal, D. Hakkani-Tür, G. Hwang, N. Michel, E. King, and R. Prasad. Advancing the state of the art in open domain dialog systems through the alexa prize. *2nd Proceedings of the Alexa Prize*, 2018. URL https://arxiv.org/pdf/1812.10757.pdf.

[28] O. Kobza, J. Čuhel, T. Gargiani, D. Herel, and P. Marek. Alquist 5.0: Dialogue trees meet generative models. a novel approach for enhancing socialbot conversations. In *Alexa Prize SocialBot Grand Challenge 5 Proceedings*, 2023. URL https://www.amazon.science/publications/alquist-5-0-dialogue-trees-meet-generative-models-a-novel-approach-for-enhancing-socialbot-conversations.

[29] A. Kumar, A. Gupta, J. Chan, S. Tucker, B. Hoffmeister, M. Dreyer, C. Monson, and A. Kumar. Just ask: Building an architecture for extensible self-service spoken language understanding. *ArXiv*, abs/1711.00549, 2017. URL https://arxiv.org/abs/1711.00549.

[30] C. P. Le, L. Dai, M. Johnston, Y. Liu, M. Walker, and R. Ghanadan. Improving open-domain dialogue evaluation with a causal inference model. In *Proceedings of the International Workshop on Spoken Dialog Systems (IWSDS)*, 2023. URL https://assets.amazon.science/94/7b/a1c23fa84cd09e58f816a1533b41/improving-open-domain-dialogue-evaluation-with-a-causal-inference-model.pdf.

[31] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.703. URL https://aclanthology.org/2020.acl-main.703.

[32] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized BERT pre-training approach, 2019. URL https://arxiv.org/abs/1907.11692.

[33] L. Mo, H. Gong, S. Singh, C.-Y. Tai, T. Zang, T. Zhang, and H. Sun. Taco 2.0: A task-oriented dialogue system with mixed initiatives and multi-modal interaction. In *Alexa Prize TaskBot Challenge 2 Proceedings*, 2023. URL https://www.amazon.science/alexa-prize/proceedings/taco-2-0-a-task-oriented-dialogue-system-with-mixed-initiatives-and-multi-modal-interaction.

[34] 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. URL http://proceedings.mlr.press/v139/radford21a/radford21a.pdf.

[35] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 1(2):3, 2022. URL https://arxiv.org/abs/2204.06125.

[36] N. Reimers and I. Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China, Nov. 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1410. URL https://aclanthology.org/D19-1410.

[37] A. Salvador, N. Hynes, Y. Aytar, J. Marin, F. Ofli, I. Weber, and A. Torralba. Learning cross-modal embeddings for cooking recipes and food images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3020–3028, 2017. URL http://pic2recipe.csail.mit.edu/im2recipe.pdf.

[38] C. Samarinas, P. Promthaw, R. Lekhwani, S. Mysore, S. M. Huang, A. Nijasure, H. Zeng, and H. Zamani. Marunabot v2: Towards end-to-end multi-modal task-oriented dialogue systems. In *Alexa Prize TaskBot Challenge 2 Proceedings*, 2023. URL https://www.amazon.science/alexa-prize/proceedings/marunabot-v2-towards-end-to-end-multi-modal-task-oriented-dialogue-systems.

[39] V. Sanh, A. Webson, C. Raffel, S. Bach, L. Sutawika, Z. Alyafeai, A. Chaffin, A. Stiegler, A. Raja, M. Dey, M. S. Bari, C. Xu, U. Thakker, S. S. Sharma, E. Szczechla, T. Kim, G. Chhablani, N. Nayak, D. Datta, J. Chang, M. T.-J. Jiang, H. Wang, M. Manica, S. Shen, Z. X. Yong, H. Pandey, R. Bawden, T. Wang, T. Neeraj, J. Rozen, A. Sharma, A. Santilli, T. Fevry, J. A. Fries, R. Teehan, T. L. Scao, S. Biderman, L. Gao, T. Wolf, and A. M. Rush. Multitask prompted training enables zero-shot task generalization. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=9Vrb9D0WI4.

[40] H. Shi, L. Ball, G. Thattai, D. Zhang, L. Hu, Q. Q. Gao, S. Shakiah, X. Gao, A. Padmakumar, B. Yang, C. Chung, D. Guthy, G. Sukhatme, K. Arumugam, M. Wen, O. Ipek, P. Lange, R. Khanna, S. Pansare, V. Sharma, C. Zhang, C. Flagg, D. Pressel, L. Vaz, L. Dai, P. Goyal, S. Sahai, S. Liu, Y. Lu, A. Gottardi, S. Hu, Y. Liu, D. Hakkani-Tür, K. Bland, H. Rocker, J. Jeun, Y. Rao, M. Johnston, A. Iyengar, A. Mandal, P. Natarajan, and R. Ghanadan. Alexa, play with robot: Introducing the first alexa prize simbot challenge on embodied ai. In *Alexa Prize SimBot Challenge Proceedings*, 2023. URL https://www.amazon.science/alexa-prize/proceedings/alexa-play-with-robot-introducing-the-first-alexa-prize-simbot-challenge-on-embodied-ai.

[41] A. Sicilia and M. Alikhani. Learning to generate equitable text in dialogue from biased training data. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2898–2917, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.163. URL https://aclanthology.org/2023.acl-long.163.

[42] A. Sicilia, Y. Asano, K. Atwell, Q. Cheng, D. Gupta, S. Hassan, M. Inan, J. Nwogu, P. Sharma, and M. Alikhani. Isabel: An inclusive and collaborative task-oriented dialogue system. In *Alexa Prize TaskBot Challenge 2 Proceedings*, 2023. URL https://www.amazon.science/alexa-prize/proceedings/isabel-an-inclusive-and-collaborative-task-oriented-dialogue-system.

[43] S. Soltan, S. Ananthakrishnan, J. FitzGerald, R. Gupta, W. Hamza, H. Khan, C. Peris, S. Rawls, A. Rosenbaum, A. Rumshisky, C. S. Prakash, M. Sridhar, F. Triefenbach, A. Verma, G. Tur, and P. Natarajan. Alexatm 20b: Few-shot learning using a large-scale multilingual seq2seq model, 2022. URL https://assets.amazon.science/f9/e2/02edd1a74115b20d356f89517f11/alexatm-20b-paper.pdf.

[44] A. Tabassum, M. Wahed, T. Yu, A. B. Mbakwe, M. Ogunleye, and I. Lourentzou. Plan-bot: Contextualized and knowledge-grounded multimodal taskbot. In *Alexa Prize TaskBot Challenge 2 Proceedings*, 2023. URL https://www.amazon.science/alexa-prize/proceedings/plan-bot-contextualized-and-knowledge-grounded-multimodal-taskbot.

[45] R. Taori, I. Gulrajani, T. Zhang, Y. Dubois, X. Li, C. Guestrin, P. Liang, and T. B. Hashimoto. Stanford alpaca: An instruction following llama model, 2023. URL https://github.com/tatsu-lab/stanford_alpaca.

[46] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, D. Bikel, L. Blecher, C. C. Ferrer, M. Chen, G. Cucurull, D. Esiobu, J. Fernandes, J. Fu, W. Fu, B. Fuller, C. Gao, V. Goswami, N. Goyal, A. Hartshorn, S. Hosseini, R. Hou, H. Inan, M. Kardas, V. Kerkez, M. Khabsa, I. Kloumann, A. Korenev, P. S. Koura, M.-A. Lachaux, T. Lavril, J. Lee, D. Liskovich, Y. Lu, Y. Mao, X. Martinet, T. Mihaylov, P. Mishra, I. Molybog, Y. Nie, A. Poulton, J. Reizenstein, R. Rungta, K. Saladi, A. Schelten, R. Silva, E. M. Smith, R. Subramanian, X. E. Tan, B. Tang, R. Taylor, A. Williams, J. X. Kuan, P. Xu, Z. Yan, I. Zarov, Y. Zhang, A. Fan, M. Kambadur, S. Narang, A. Rodriguez, R. Stojnic, S. Edunov, and T. Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023. URL https://ai.meta.com/research/publications/llama-2-open-foundation-and-fine-tuned-chat-models/.

[47] R. University. Diwbot: A cooking and diy conversation guidance system. In *Alexa Prize TaskBot Challenge 2 Proceedings*, 2023. URL https://www.amazon.science/alexa-prize/proceedings/diwbot-a-cooking-and-diy-conversation-guidance-system.

[48] R. Valerio, J. Bordalo, M. Yarom, Y. Bitton, I. Szpektor, and J. Magalhaes. Transferring visual attributes from natural language to verified image generation. *arXiv preprint arXiv:2305.15026*, 2023. URL https://arxiv.org/abs/2305.15026.

[49] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, C. Ma, Y. Jernite, J. Plu, C. Xu, T. Le Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush. Transformers: State-of-the-Art Natural Language Processing. pages 38–45. Association for Computational Linguistics, Oct. 2020. URL https://www.aclweb.org/anthology/2020.emnlp-demos.6.

[50] R. H. Zhang, P. Sell, Y. Zhang, L. Che, A. G. S. K. S, R. Bhatt, P. Nagasubramaniam, S. V. S. Dave, H. Maniar, V. Dasu, and R. Zhang. Evoquerbot: A multimedia chatbot leveraging synthetic data for cross-domain assistance. In *Alexa Prize TaskBot Challenge 2 Proceedings*, 2023. URL https://www.amazon.science/alexa-prize/proceedings/evoquerbot-a-multimedia-chatbot-leveraging-synthetic-data-for-cross-domain-assistance.

[51] S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. Diab, X. Li, X. V. Lin, T. Mihaylov, M. Ott, S. Shleifer, K. Shuster, D. Simig, P. S. Koura, A. Sridhar, T. Wang, and L. Zettlemoyer. Opt: Open pre-trained transformer language models, 2022. URL https://arxiv.org/pdf/2205.01068.pdf.

[52] K. Zheng, J. Bheemanpally, B. Garg, S. Heo, D. Sonawane, W. Chen, S. V. S, and X. E. Wang. Sage: A multimodal knowledge graph-based conversational agent for complex task guidance. In *Alexa Prize TaskBot Challenge 2 Proceedings*, 2023. URL https://www.amazon.science/alexa-prize/proceedings/sage-a-multimodal-knowledge-graph-based-conversational-agent-for-complex-task-guidance.

[53] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. P. Xing, H. Zhang, J. E. Gonzalez, and I. Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023. URL https://arxiv.org/abs/2306.05685.