PLAN-Bot: Contextualized and Knowledge-Grounded Multimodal Taskbot

Afrina Tabassum
Virginia Tech
afrina@vt.edu

Muntasir Wahed
Virginia Tech
mwahed@vt.edu

Tianjiao Yu
Virginia Tech
tianjiao@vt.edu

Amarachi Blessing Mbakwe
Virginia Tech
bmamarachi@vt.edu

Makanjuola Ogunleye
Virginia Tech
mogunleye@vt.edu

Ismini Lourentzou
Virginia Tech
ilourentzou@vt.edu

Abstract

We introduce PLAN-Bot, a knowledge-grounded multimodal taskbot to guide users through completing real-world cooking or do-it-yourself (DIY) tasks in the Alexa TaskBot Challenge 2. A successful task-driven conversational assistant must be effective and engaging, capable of assisting users from task discovery to providing step-by-step instructions for the chosen task. To achieve this, PLAN-Bot incorporates several key features. Firstly, PLAN-Bot is equipped with a robust and adaptable query extraction system to efficiently search for specific tasks or suggest interesting and seasonal activities to the users. Secondly, each task is represented using a hierarchical graph, enabling organized and seamless navigation for users throughout the process. Additionally, PLAN-Bot can answer contextual inquiries related to the selected task by using a knowledge-grounded question-answering module that ensures users receive accurate and informative responses to their questions. Furthermore, we propose the use of fine-grained recipe embeddings, enabling improved cross-modal retrieval tasks and ingredient substitution, and enhancing the overall user experience. To prioritize user safety, PLAN-Bot integrates a robust safety classifier that prevents the bot from providing harmful advice, resulting in more than 98% uptime during the semifinal interaction period. As of June 30, 2023, PLAN-Bot has achieved 3.5 and 3.58 ratings in terms of L7d (↑ .27) and L14d (↑ .18), respectively, while also maintaining a 44.2% (↑ 8.8) completion rate and a conversation resume rate of 19.07%.

1 Introduction

The emergence of task-oriented conversational agents has significantly improved daily life experiences for millions of users. Among these agents, Alexa stands out as one of the most widely used, guiding millions of users through various real-world tasks. However, domains such as cooking and home improvement pose challenges due to their task complexity and dependency among steps. In the Alexa Prize Taskbot Challenge 2 [Agichtein et al., 2023], we propose PLAN-Bot, a contextualized and knowledge-grounded multimodal taskbot focused on empowering users to effortlessly complete tasks in these domains. We envision PLAN-Bot as an indispensable tool for all Alexa users, with special consideration for those with disabilities, ensuring inclusivity in its design. To achieve this vision, we incorporate several cutting-edge features that highlight the vast capabilities of our taskbot, catering

2nd Proceedings of Alexa Prize TaskBot (Alexa Prize 2023).

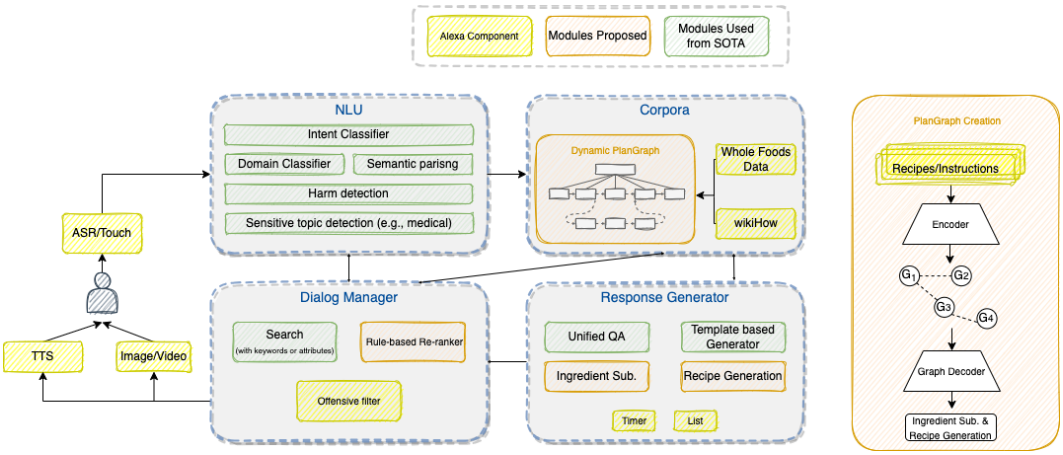

Figure 1: Overall System Design

to the diverse needs of users across multimodal and headless devices. Our data-centric architecture allows engaging and smooth interaction between PLAN-Bot and its users, making it an essential and reliable companion for completing their tasks. This technical report provides a comprehensive overview of the proposed PLAN-Bot and delves into key features contributing to its user-friendly nature and indispensability. The core contributions of this work encompass the following:

(1) PlanGraph: We introduce a novel hierarchical PlanGraph that intelligently extends the task structure, enabling PLAN-Bot to support users in handling complex and diverse tasks quickly and efficiently.

(2) Fine-Grained Recipe Embeddings: PLAN-Bot employs procedural recipe embedding techniques to facilitate image retrieval for tasks, enriching the user experience.

(3) Robust Safety Classifier: Ensuring user safety is our top priority. To achieve this, PLAN-Bot employs a robust text classifier that effectively identifies and refrains from providing harmful, legal, medical, or financial advice to users.

(4) Task-Related Question Answering: The PLAN-Bot is equipped with task-related question-answering capabilities for seamless user interaction, enabling it to respond promptly and accurately to user inquiries.

2 System Overview

2.1 Architecture

PLAN-Bot (Figure 1) is developed based on the CoBot framework [Khatri et al., 2018]. At each turn, a user can interact with our system by speaking or using the touch screen (if available). On the user end, Alexa's Automatic Speech Recognition (ASR) transcribes the user's speech into a text utterance. In the touch screen mode, the touch event is described as a set of argument-value pairs in Alexa Presentation Language (APL).

Upon receiving the user's input, the PLAN-Bot system first passes the user utterance through the Natural Language Understanding (NLU) unit. The NLU unit subsequently fetches other modules, hosted as remote EC2 instances. To fulfill latency requirements, most modules run in parallel and generate flags as their outcomes, which are checked when the final output is ready to be returned. After the successful retrieval of results from the NLU pipeline, our system invokes the Dialogue Manager (DM) module. DM relies on both state attributes and intent classification results to control the dialogue flow, handle exceptions, and guide the conversation toward task completion. The dialogue management unit is equipped with a specialized output re-ranker, designed to guarantee the selection of the most suitable responses. Lastly, DM decides on which response generation modules to execute.

In terms of the Response Generator module, the system has template-based response generators and neural response generators. The neural module can be viewed as a question-answering module to address users' questions, while the template-based generator serves as a complementary component responsible for generating more structured and consistent responses. By leveraging the strengths of both approaches, our system achieves a fine balance between flexibility and precision in its responses, catering to a wide range of user inquiries while maintaining coherence and accuracy. Another advantage of the neural response generator is addressing in-context questions based on selected recipes and instructions.

When the DM decides that the question pertains to a specific task, it searches in the Whole Foods and wikiHow databases to gather relevant contextual references. Then, the corresponding recipes and instructions will be embedded into PlanGraph, which enables the response generator to suggest the next instruction or ingredient/tool substitution and in the future, generate a modified task. The final textual or multimodal response is rendered by Alexa's SSML Text-To-Speech (TTS) and APL services before being delivered to the user. To guarantee ethical integrity, safety filters are deployed both for inputs and outgoing responses at various points.

Navigation and Utilities. At present, our bot supports the following commands: `More Results` to show more search results, `More/Less Details` for showing/hiding details as per user request, `Navigation` such as next/previous step, go to the ingredients/'X' step/last step, `Repeat` for repeating the last step, `Complete` as an indication of completing a task, `Start a New Task` to start a new task in the middle of another task, and `Stop` to stop the bot any time. In addition, we support Alexa list and timer management features using the `List` and `Timer` intents, respectively, which detect a diverse set of utterances and are accumulated by analyzing the Alexa user conversation data.

3 Modules

Text classification is an essential module for a multitude of functionalities that PLAN-Bot supports, *e.g.*, identifying harmful, financial, medical, legal, dangerous text, parsing search queries *etc.* Although instruction-tuned large language models (LLMs) perform exceptionally on text classification tasks in our experiments, it is expensive to run an LLM on remote modules. For example, inference with a FlanT5XL [Chung et al., 2022] model on an A100 GPU takes around 0.75 seconds on average, which is not acceptable in a real-time conversation setting. Consequently, we adopt a hybrid approach to text classification. For the more complicated tasks, *e.g.*, the harmful text classifier and the financial, legal, and medical text classifier, we finetune a much smaller RoBERTa [Liu et al., 2019] model on the classifications from LLMs. For other cases, *i.e.*, request for games, team identification texts, *etc.*, we use the deterministic text classification module described in Section 3.1. Our language models, along with the deterministic text classification module, have resulted in a robust and efficient text classification pipeline, leading to 98% up-time in the beta stage, 96% up-time in the GA stage, and 98% up-time in the semifinals.

Harmful Text. To train the harmful classifier, we first collect a corpus of task titles from wikiHow. Then, we employ an instruction-tuned Flan-XL model to generate labeled examples using the following prompt: "Harmful tasks are tasks that can cause harm or injury to oneself or others if not performed properly or with the necessary precautions. For example, "shooting a gun", "breaking the speed limit", and "setting a house on fire" are some examples of harmful tasks. A user asks how to make a paper boat? Question: Is the user asking about a Harmful task? Answer by giving step-by-step reasoning (Yes/No):" We end up with $111,264$ non-harmful and $1,113$ harmful task titles. We subsequently fine-tune a RoBERTa [Liu et al., 2019] model on this data with 90-10 training-test split, which achieves 89% recall (on harmful tasks) on the test set.

Financial, Legal, and Medical Text. Ensuring customers' safety and safeguarding Amazon's reputation is of utmost importance. Given this, it is crucial to refrain from providing financial, legal, and medical advice. To address this challenge effectively, we have developed a robust

Financial-Legal-Medical (FML) domain classifier capable of accurately classifying user and PLAN-Bot text. The FML classifier consists of six classes: Financial, Medical, Legal, Cooking, DIY, and General. Each category represents a specific domain that the classifier can accurately identify and distinguish. To train the FML classifier, we have collected a diverse dataset from various reliable sources to build this classifier. Specifically, we gather financial questions from FinQA [Chen et al., 2021], medical questions from MedQuAD [Ben Abacha and Demner-Fushman, 2019], cooking and DIY-related questions from Wizard of Tasks [Choi et al., 2022], general questions from ComQA [Abujabal et al., 2019], and legal questions from the Avoo website [Avvo.com, 2023]. This diverse dataset enables us to cover various topics and create a classifier with broader applicability. We partition the dataset into train, validation, and test sets. The training set comprises $4,800$ samples, while the validation and test sets contain $1,200$ and $1,500$ samples, respectively. We achieve strong performance across the different classes (Financial, Medical, Legal, Cooking, DIY, and General) by fine-tuning the RoBERTa model on our annotated dataset. Fine-tuning a small RoBERTa model requires fewer computational resources but reaches comparable performance to more recent larger language models evaluated in zero-shot settings. The model achieves an overall accuracy of 97%, showcasing its ability to make precise and reliable predictions. We recently chose to omit this remote module due to resource constraints (budget limitations and latency requirements) and instead leverage the Amazon-provided domain classifier that also exhibits excellent performance. Nevertheless, our robust FML domain text classifier served as a valuable tool in the initial phases, ensuring that customers receive appropriate responses and assistance while upholding system integrity and enhancing the conversational experience of customers.

3.1 Deterministic Text Classification

We experiment with two different versions of the deterministic text classification module. In the first version, we use a set-based approach to classify texts based on partial matches. In the second version, we use the Aho-Corasick algorithm [Aho and Corasick, 1975] to efficiently retrieve matches. However, we observe that the Aho-Corasick algorithm does not yield significant performance improvements because of the relatively small number of patterns for the different text classification tasks. For example, the set-based implementation only takes 0.0012 milliseconds on average to find a text from a set of 1000 patterns. Consequently, for simplicity, we have currently deployed the set-based version.

We use the deterministic text classification module for a multitude of tasks, for example, detecting whether the user is asking to play a game or a team identification question, adjustable search query generator, *etc.* For the detection of the game or team identification texts, we first collect data from our team members. More specifically, we ask each member to act as an adversarial user trying to make the bot do something (*e.g.*, play games) that it is not supposed to do. Then, we expand the list of such user utterances using GPT 3.5 to generate more similar texts. For game text detection, we additionally compile a set of 50 popular online conversation-based games and replace such occurrences with a special token.

3.2 Noun Phrase Detection

Noun-phrase detection is a fundamental component for various natural language processing tasks and information extraction systems [Gordon et al., 2004, Lin et al., 2012]. By identifying and extracting noun phrases from textual data, we gain valuable insights into the underlying structure and meaning of sentences. Noun phrases serve as key elements in grammatical constructions, often encapsulating essential information about entities, objects, or concepts in a sentence. For example, for query extraction, we need to extract the search terms (mostly recipe names and DIY task names) from natural sentence inputs so that we can retrieve the inquired recipes or instructions from the Whole Foods and WikiHow datasets. Initially, we address this problem by employing Spacy [Honnibal et al., 2020] and NLTK [Loper and Bird, 2002] for Part-Of-Speech tagging (POS) and dependency parsing. Once the words are tagged, we proceed to eliminate stop words and focus on verbs, nouns, and noun phrases. The analysis of the search results revealed that incorporating both verbs and nouns leads to an enhancement task search. However, for recipe results, the improvement is observed by considering solely nouns and certain cooking-related verbs (*e.g.*,

"grill" and "fry"). This strategic approach enables us to efficiently search within datasets by filtering out irrelevant terms, significantly boosting the performance of the search pipeline. State-of-the-art language models are capable of noun phrase detection and often produce more accurate results. However, after testing models such as RoBERTa [Liu et al., 2019], InstaFoodRoBERTa [Egli, 2023], *etc.*, and achieving up to 91% F1 score, we observe that the processing time of these models usually falls between two to four seconds, which is unsatisfactory for the overall system latency requirement.

3.3 Adjustable Search Query Generator

In order to support users with various dietary restrictions, we have implemented a novel Search Query Generator. Using the text classification module described in Section 3.1, we employ a deterministic approach to let users further adjust their search queries. Currently, we support adjusting search queries with the occasion, cuisine, meal type, and ingredients. For example, a user can start the search with cookie recipes for Mother's Day. Then, they can filter the search by saying "show me recipes that do not contain eggs" or "show me recipes with almonds". The list of occasions, cuisines, and meal types are based on the available options in the Whole Foods Market API, while the ingredient query extraction relies on the Noun Phrase Detection module described in Section 3.2.

3.4 Intent Classification

To perform intent classification, we initially finetune a pre-trained RoBERTa [Liu et al., 2019] model on the Wizard of Tasks dataset [Choi et al., 2022], which has 549 conversations and 18K utterances. In our experiments, we achieve an 80% accuracy on the test set. However, the performance was heavily imbalanced across different classes. For example, while the model achieved 94% accuracy for the "request next step" intent class, the accuracy for "chitchat" and "ask student question" were 42% and 31%, respectively. One reason behind this is the imbalanced nature of the data. There are $3,758$ instances for "request next step", whereas there are only 120 and 130 for "chitchat" and "ask student question", respectively. At the same time, this may also be due to the fact that "chitchat" and "questions" can encompass a more diverse set of natural language conversations compared to "next step".

In real-life interactions with the bot, we observe that a more deterministic approach to routing to the correct responder is more accurate. This approach also provides more control over the continuous evolution of the bot based on previous user interactions. Therefore, we currently use a logic-based deterministic approach to classify user intents. We understand that this is limited to the known ways of user interaction and can not adapt to the whole set of natural language conversations. In the future, we plan to continue our work on improving the intent classification model and switch to a hybrid approach.

3.5 Online Dynamic Cache

Through our analysis of user conversations, we observe that many users search for similar things. We hypothesize that this can be due to two reasons. Firstly, many users follow the botś suggestion and search for one of the suggested tasks. Secondly, some tasks are inherently more popular than others and users are more interested in those, for example, pizza, chicken, *etc.* In Figure 2, we observe that 75.10% and 51.91% of the search queries appear more than once, respectively for recipe and DIY tasks. Additionally, in our experience, we find that the search query APIs can sometimes be unreliable due to a variety of reasons, for example, server load, bandwidth constraints, *etc.*, resulting in a lot of timeout errors, especially during busy hours. Consequently, given the highly time-sensitive nature of real-life conversations, we implement a caching mechanism to serve search queries faster to users.

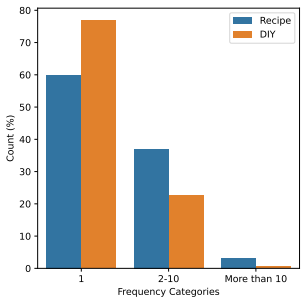

Figure 2: Frequency of different search queries during the semifinals.

More specifically, we maintain a DynamoDB table to keep track of different search queries, their frequencies, and the last time a user searched for that particular query. If a query is found in the cache, we extract the response from DynamoDB, otherwise, we send a request to one of the search APIs. Extracting an existing item takes only around 0.15 seconds on average, resulting in fewer timeout errors and an overall smoother experience for users. To ensure that the DynamoDB table does not exceed a maximum size (500 entries), we eliminate entries from the table based on a certain threshold which is set using the count and last access time of a particular entry.

3.6 Task-related Question Answering

We integrate a Question-Answering (QA) model [Khashabi et al., 2020], which can handle user questions that are context-based, task-specific, and not generic. The objective is to employ state-of-the-art QA techniques to generate the most suitable context-aware answer for a given user question. For example, if a user asks a question about a recipe, we would like to present to the user an answer that is extracted from the recipe instructions or ingredients. Consequently, in the backend, the context supplied to our QA model is an aggregation of the recipe instructions and ingredients. The model then generates an answer from this context.

We relied on the robust UnifiedQA[Khashabi et al., 2020, 2022] model for this task. The UnifiedQA is a single pre-trained question-answering model trained on a diverse set of question-answering datasets and formats. This allows the model to be format and data-agnostic. To further improve QA model performance, we finetune UnifiedQA on two cooking and DIY datasets. The first dataset is Amazon's Wizard of Task (WOT) dataset [Choi et al., 2022] that contains conversations from Conversational Task Assistants (CTA). We parse this dataset and derive a new one containing the aggregated instructions and ingredients as context, the student/user's question, and the teacher/bot's answer. The second dataset is the publicly available DoQA dataset [Campos et al., 2020], a dataset with $2,437$ dialogues and $10,917$ QA pairs collected from three Stack Exchange sites using the Wizard-of-Oz (WoZ) method with crowdsourcing [Green and Wei-Haas, 1985, Hu et al., 2023]. In the WoZ simulation, a user engages in interactive communication with a person called Wizard, designed to mimic a system. The resulting communication can serve as a dataset for training dialog models or can be used to improve user interfaces. For the DoQa dataset, the simulation involves two crowd workers assuming distinct roles. The first worker acts as the user who poses questions related to a specific topic or discussion on Stack Exchange. The second worker adopts the role of a domain expert who leverages responses and information from the Stack Exchange topic as context to provide well-informed answers to the user's question. In this dataset, QA pairs and dialogues are in 3 different domains (cooking, travel, movies). For the scope of the Alexa TaskBot Prize Challenge, we extract only QA pairs on the cooking domain. We combine both derived datasets and finetune the UnifiedQA model on the aggregated data. Our fine-tuned UnfiedQA model can provide answers to context-based cooking and DIY questions with a consistent confidence score that is above 90%.

3.7 Open-ended Question Answering

While working on a task, a user might ask a question whose answer is not in the task document. For example, when performing the task "How to make berry chantilly cake?", a user might ask "what is chantilly cake?". To answer these open-ended questions, we utilize Amazon's EVI API[1]. Next, in order to filter out irrelevant responses, we use the QA factoid binary classification model provided by Amazon, which classifies an answer as factoid or non-factoid. While answering open-ended questions, we prioritize providing answers that are 100% factually correct. This high level of precision is essential because even a few incorrect responses can compromise the successful completion of the task.

3.8 BERT Ranker Response Selector

In order to facilitate accurate and high throughput of our PLAN-Bot responder system, we incorporate Alexa Cobot's BERT-based response selector model [Cobot, 2023]. We

[1]https://en.wikipedia.org/wiki/Evi_(software)

refrain from using a rule-based ranking strategy which can get cumbersome as the number of responders increases. The BERT Ranker selects the most appropriate and relevant responder from a list of responders. To use the BERT Ranker, we send an API request through the Alexa Prize Cobot toolkit service. The API request consists of a dialog context that contains a list of past conversation turns between the bot and the user. Each conversation turn consists of a tuple of (text, response, and selected responder). The text represents the user input in this conversation turn, while the response represents the bot response and the selected responder represents the responder that was eventually selected. Utilizing the Bert Ranker initially seemed promising with the expectation of faster and more accurate responses for our dialogue system. However, deployment latency and resource constraints led us to transition to a deterministic rule-based ranking approach.

4 Enhancing PLAN-Bot

4.1 PlanGraph

Task Representation. We develop PlanGraph, a new hierarchical task structure to represent diverse and complex tasks. PlanGraph is a directed graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ where $\mathcal{V}$ is the set of nodes, denoted as $\mathcal{V} = \{s_1, s_2, \ldots, s_n\}$ representing the attainable goals or steps in the task and $\mathcal{E}$ is the set of directed edges denoted representing the dependencies between nodes. An edge $e_{ij} \in \mathcal{E}$ implies that node s_i must be completed before starting node s_j. Each node s_i in PlanGraph contains the following step attributes:

- Title (denoted as T_{s_i}): The title or name of the task step s_i.
- Description (denoted as D_{s_i}): A textual description elaborating on the details and instructions for completing step s_i.
- Image (denoted as I_{s_i}): An optional image associated with step s_i.
- Video (denoted as V_{s_i}): An optional video linked to step s_i for further assistance.
- Ingredients (denoted as R_{s_i}): A list of ingredients required for recipe-related tasks.
- Tools (denoted as U_{s_i}): A list of tools or equipment necessary for DIY-related tasks.
- Other information (denoted as O_{s_i}): Additional task-specific attributes and data related to step s_i.

PlanGraph Creation. At present, we follow a retrieval-based approach in order to create PlanGraph. In the retrieval-based approach, we utilize the Whole Foods Market API and wikiHow API provided in the Cobot architecture to retrieve tasks with similar titles with the search query, and present at most three most similar results to the user. Once a user selects a task, we create the PlanGraph from the retrieved document. Firstly, we create a node representing the goal. Then, we create the children nodes of the goal node, which represent each instruction with edges from and to preceding and following instructions, respectively.

4.2 Ingredient Substitution

The potential benefits of recipe personalization through ingredient substitution are vast, as it can aid individuals in meeting their specific dietary needs and preferences, while also avoiding allergens, thereby promoting a more inclusive culinary exploration in every kitchen. There exists an ingredient substitution dataset, Recipe1MSubs [Fatemi et al., 2023], consisting of 110467 pairs of ingredient substitutions. However, it is automatically extracted from user comments on recipe websites and contains many incorrect substitutions. To address the persistent challenge of misalignment, we leverage the recipe embedding and large language models to further filter the dataset. We extract 24679 unique substitution pairs in total, enabling accurate ingredient substitution for the users.

4.3 Visuals for Task Steps

Representing procedural text for cross-modal retrieval poses inherent challenges, particularly when factoring in the compositional intricacies involved in recipe image retrieval. This task

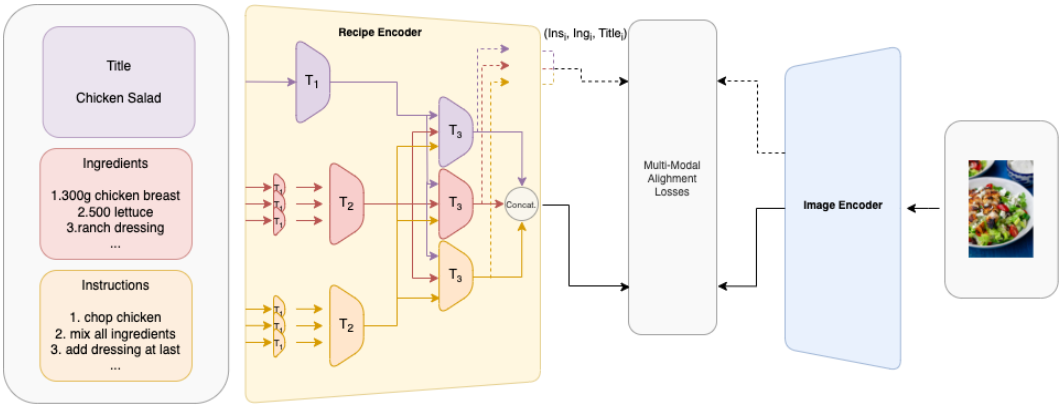

Figure 3: Fine-Grained Cross-Modal Recipe Encoder Overview

has been tackled using classification [Chen et al., 2017a] and cross-modal learning [Wang et al., 2019, Carvalho et al., 2018] techniques. In classification-based methods, food images are annotated with rich attributes such as ingredients, cooking methods, and cutting techniques. These attributes are then matched against words extracted from recipes to facilitate retrieval. However, a significant drawback of this approach is the extensive effort required in manually labeling food attributes. In contrast, cross-modal learning offers a more efficient solution by training a latent space that can accommodate both image and text modalities, enabling measuring of similarity between recipe components and significantly reducing labeling efforts [Chen et al., 2017b]. Although efficient, cross-modal learning is inherently unexplainable [Zhu et al., 2019]. Unlike classification-based approaches, cross-modal learning is unable to list out the matched attributes as evidence to recount the retrieval results.

Given a recipe embedding model, we retrieve original or similar illustrative images for single steps of a recipe. Our goal is to enhance search results by presenting explanatory images related to recipes. To address the challenge of understanding the semantics of instruction texts, we have developed a cross-modal recipe embedding model that performs fine-grained alignment of recipe component encodings, including titles, ingredients, and instructions, within a shared representation space alongside corresponding image embeddings. As shown in Figure 3, to capture the interdependent relation between the textual components, we use a model with multiple Transformers hierarchically, inspired by [Shukor et al., 2022]. To improve fine-grained alignment within the various recipe components, we introduce a new loss function that captures the hierarchical structure inherent in recipe classes. The proposed cross-modal understanding model is trained on the Recipe1M dataset [Salvador et al., 2017]. Using this model, we perform a direct cosine similarity comparison between step sentences and retrieved images. Our evaluation on the RecipePlan dataset [Lu et al., 2023] yielded an initial F1-score of 48%. Subsequently, fine-tuning with a linear classification head resulted in an improved F1-score of 55%. However, we opted to not include retrieved images during finals, primarily due to the overall low performance and the limited time for A/B testing.

5 Analysis

5.1 Customer Satisfaction

Overall Trend. In Figure 4, we observe that we maintain an above 3 rating for a very long period of time (June 22 - July 29). During this period, our L7d rating dropped below 3.0 only once due to some deployment issues from July 3 to July 10. Other than that, our ratings remained consistently higher, with the bot leading the leaderboard for 13 days in a row, from June 24 to July 6, 2023. More recently, as of June 30, 2023, PLAN-Bot has achieved 3.5 and 3.58 ratings in terms of L7d (↑ .27) and L14d (↑ .18) respectively. while also maintaining a 44.2% (↑ 8.8) completion rate and 19.07% resumed conversation rate.

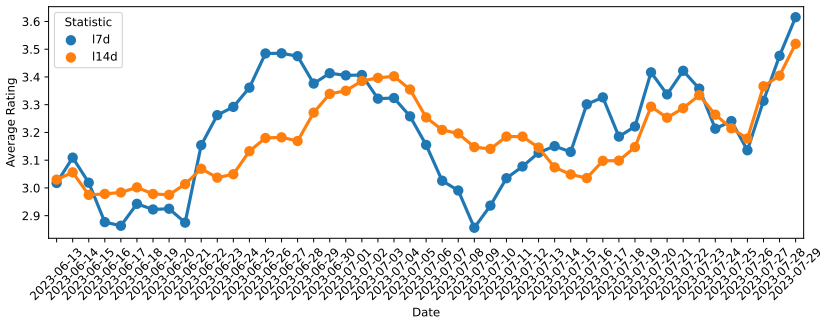

Figure 4: Customer satisfaction in terms of 7-day (L7d) and 14-day average (L14d) ratings.

Highly Polarized Rating. We observe that ratings are heavily polarized for our bot (as shown in Figure 5). Specifically, 26.66% users rated the bot 1 star, while 34.57% rated with 5 stars. This explains the high variance in our ratings. One reason behind this may be the disparity of ratings in Recipe and DIY tasks. We have observed that the mean rating for Recipe tasks is 0.6 higher than for DIY tasks.

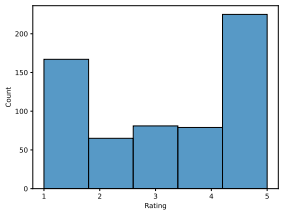

Figure 5: User ratings.

Conversation Duration. We also observe that users who spend more time with the bot tend to leave higher ratings. In Figure 6, we visualize the difference in user ratings with respect to conversation duration. We exclude conversations less than 90 seconds from this analysis as often in such cases, the users accidentally end up in the bot and leave without doing much. We observe that users who spend $90 - 110$ seconds in the bot have an average rating of 2.88, whereas users spending more than 228 seconds have provided an average rating of 3.69.

5.2 User Testing

During the beta stage, we performed user testing to understand what users are expecting from the bot. We recruited several volunteer students from Virginia Tech for the user testing. To imitate a real-life use case, we did not share too much information with the volunteers. When presenting our bot to the user, the test conductor adhered to the following instructions:

- ☑ Present the device to the user.
- ☑ Don't give too much context. Just say "This is a bot that can help you with recipe and home improvement tasks. Please try it out."
- ☑ Invoke the bot and then ask the user to continue.
- ☑ Observe the user as they try to use the bot and take notes.

The test conductor observed the user while they interacted with our bot, and took notes focusing on the following:

- ✎ What did the users try to do?
- ✎ In what cases did the bot work well?
- ✎ In what cases did the bot fail?

In the end, the users were given the option to share additional feedback, focusing on what they liked and did not like about the interaction. In total, we conducted 20 user tests, based on which we made several bot improvements in the early stages. Some of the features implemented after the user testing include the following:

- 💡 Some users did not like any of the three suggested options from the search results. In that case, they were trying to retrieve more options by saying "more options" or "no". We later implemented this feature and prompted users to say "more options".

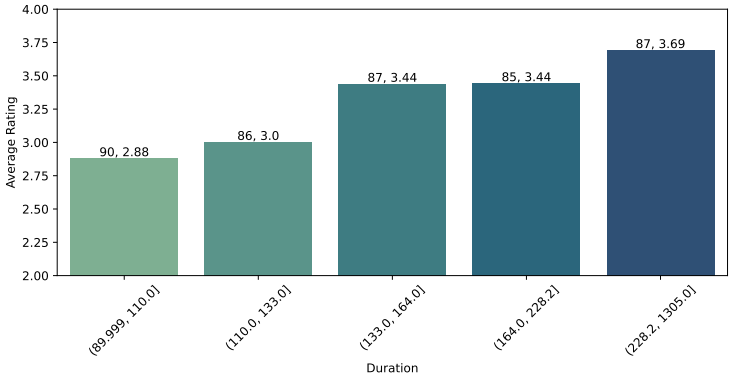

Figure 6: Customer satisfaction ratings vs. conversation duration. The x-axis represents different ranges of duration for rated conversations. Numbers displayed on top of each bar represent the number of rated conversations and the average rating in that range of duration.

- In the early stage, users were struggling to get inside a task after searching. Based on this observation, we prompted the user to say "the first option" to select a task.
- Some users were trying to start a new task before completing the current task. We implemented a feature later, where in such cases PLAN-Bot asks the user "I see that you want to start a new task. Is that correct?". If the user responds with "yes", then the bot takes them to the search query. If they negatively respond with "no", the bot takes them back to the task they were doing.
- Some users were trying to select the task by saying the name of the task. We later implemented this feature in PLAN-Bot. Now, the user can select by name, as well as enumeration such as "the first option".

5.3 Logs

During the beta stage, and to understand how users have previously interacted with the bot, we analyzed the previous year's conversation data with the objective of ensuring that the bot handles the more frequent utterances correctly. Some of the features that we implemented based on observation from the previous year's conversation include:

- **Better Navigation**: We noticed that users tried to go back to the previous step, and sometimes to go back to the n-th step. A lot of users also tried to list all the ingredients again after starting the task. We have implemented all these features based on our analysis and observations.
- **Different ways to interact**: Some users tried to select a task by saying "the first option" while some others tried to select using "option one", "one", *etc.* Our bot effectively supports all these different utterances.

Our bot currently handles more than 95% of the top-200 utterances correctly, which encompasses 84% of the total user utterances in the dataset. Some of the utterances that we can not handle correctly include "computer" (maybe this utterance indicates a different way to say Alexa?) and "summary". Additionally, we proactively keep track of recent conversations to understand any bot failure cases and subsequently perform any corresponding improvements. For example, we have recently observed that a user was trying to filter a recipe query using ingredients by saying "show me options without eggs". We deservedly got a 1-star rating from the user, but we soon after implemented this feature, and now the bot supports filtering queries by both including and excluding certain ingredients. Similarly, based on feedback, we have recently observed the disparity between our bot's performance in cooking and DIY tasks was 0.6 on average during the semifinal period. We have taken action by implementing a few features in the bot, including the switch to detail view from the list view for the wikiHow tasks. Similarly, we have noticed that some users were annoyed by the long speech of wikiHow steps. Consequently, we have implemented the option to switch between more details and fewer details for wikiHow tasks.

5.4 Themed Events

During the past few months, we have celebrated three themed events as part of the Alexa Prize Taskbot Challenge. The first one was the baking week, which ran from May 22 to May 28, 2023. We had a significant drop in per-day users during this baking-themed event. However, our ratings improved during the event, with an average rating of 3.24, compared to 2.74 and 2.91 respectively before and after the event (until June 8). After the event, we continued to provide users with task suggestions, which potentially contributed to a higher rating. We observed a significant increase in both the percentages of completed tasks and returning users during the baking-themed event. Our percentage of completed tasks was 40.54% during the event, compared to 35.05% and 34.41% before and after the event. Similarly, our returning users were 24.32% during the event, compared to 10.31% and 18.27% before and after the event. The second event was summertime, which ran from June 26 to July 5, 2023. Unlike the baking-themed event week, we did not observe a similar uplift in rating during this event, with the average rating dropping from 3.36 to 3.3. Upon closer inspection, we noticed that users searching for DIY tasks increased significantly during this event and our bot was under-performing in the DIY tasks. Based on this observation, we later made several improvements in DIY tasks, as described in Section 5.3.

6 Lessons Learned

Guide the users. In our user testing, several users reported that they felt lost when they entered the bot. In follow-up questions, they revealed that they did not have any specific tasks in mind and when the bot asked "What can I help you with?" they again felt lost. Some users also reported that it was not straightforward how to navigate the bot, *i.e.*, select a task, move forward once inside the task, *etc.* We have also noticed users getting frustrated in the beta stage due to a lack of direction. This lack of direction is also true for users who invoke the bot just to try something, while not knowing completely what the bot is capable of. In some cases, we have noticed that users try irrelevant conversations, for example, general chitchat, trying to play music, or games, *etc.* Consequently, we have made several improvements in the bot. Firstly, we have implemented a recommendation phase which always suggests the user some tasks. Secondly, we have implemented hints at different parts of the bot to help the user navigate, for example, "to go to the next step, say next", "if you are ready, say start cooking", *etc.*

No "right way". In the initial phase, we assumed that the users would follow a standard way of interacting with the bot. But we have found that not to be the case. For example, upon hearing the first step, there are a lot of different prompts users may say to move forward despite there being a hint. Some users say "okay" while some users follow a more real-life conversation setting such as "what should I do next" or "can we move forward?" Similarly, once presented with the results for a search query, while most users follow the "the first one" hint from the bot, a lot of users try to select the recipes using other utterances such as "option two", "last one", *etc.* Other users prefer to use recipe names to select a recipe. In later stages, we have made an attempt to consistently look at how users are interacting with the bot and improving the bot based on the interactions.

Long speech. During user testing, we have noticed that prolonged, monotonous bot responses tend to disengage most users. This is especially true for wikiHow DIY tasks, where the steps often exceed 5 sentences. At the same time, we have also observed a notable discrepancy in user ratings between those performing DIY tasks and cooking tasks, with DIY users consistently giving lower ratings (approximately 0.6 difference on average in the first month of the semifinals). Consequently, to address this issue, we have implemented shorter, more concise step descriptions for the DIY tasks. At the same time, we recognize that some users may prefer more detailed instructions. Consequently, we have introduced a dynamic feature that allows users to switch between more in-depth or succinct task explanations according to their preferences. This hybrid approach strikes a balance between providing comprehensive guidance and catering to individual user needs, ultimately enhancing the overall user experience.

7 Conclusion

We propose PLAN-Bot, a multimodal contextualized TaskBot that guides users through completing complex tasks in the cooking and do-it-yourself (DIY) tasks. To enhance user experience, PLAN-Bot leverages a proposed PlanGraph for decomposing a task in a hierarchical structure, an adjustable search query generator for facilitating a user to change the query in multi-turn settings, a knowledge-grounded question-answering module to answer task-related questions. Moreover, we build a robust safety classifier to prevent giving harmful advice and fine-grained recipe embeddings to perform cross-modal image retrieval for each step and provide substitutions for ingredients. PLAN-Bot makes adaptable conversation a reality by guiding users through completing complex tasks and enabling seamless user interactions during task completion.

8 Acknowledgments

We would like to thank Amazon and the Amazon Alexa Prize team for their financial and technical support.

References

Abdalghani Abujabal, Rishiraj Saha Roy, Mohamed Yahya, and Gerhard Weikum. Comqa: A community-sourced dataset for complex factoid question answering with paraphrase clusters. In *Proceedings of NAACL-HLT*, pages 307–317, 2019.

Eugene Agichtein, Michael Johnston, Anna Gottardi, Cris Flagg, Lavina Vaz, Hangjie Shi, Desheng Zhang, Leslie Ball, Shaohua Liu, Luke Dai, Daniel Pressel, Prasoon Goyal, Lucy Hu, Osman Ipek, Sattvik Sahai, Yao Lu, Yang Liu, Dilek Hakkani-Tür, Shui Hu, Heather Rocker, James Jeun, Akshaya Iyengar, Arindam Mandal, Saar Kuzi, Nikhita Vedula, Oleg Rokhlenko, Giuseppe Castellucci, Jason Ingyu Choi, Kate Bland, , Yoelle Maarek, and Reza Ghanadan. Alexa, let's work together: Introducing the second alexa prize taskbot challenge. In *Alexa Prize TaskBot Challenge 2 Proceedings*, 2023. URL `https://www.amazon.science/alexa-prize/proceedings/alexa-lets-work-together-introducing-the-second-alexa-prize-taskbot-challenge`.

Alfred V Aho and Margaret J Corasick. Efficient string matching: an aid to bibliographic search. *Communications of the ACM*, 18(6):333–340, 1975.

Avvo.com. Avvo free legal advice dataset. Retrieved from `https://www.avvo.com/free-legal-advice`, 2023.

Asma Ben Abacha and Dina Demner-Fushman. A question-entailment approach to question answering. *BMC Bioinformatics*, 20:1–23, 2019.

Jon Ander Campos, Arantxa Otegi, Aitor Soroa, Jan Milan Deriu, Mark Cieliebak, and Eneko 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, 2020.

Micael Carvalho, Rémi Cadène, David Picard, Laure Soulier, Nicolas Thome, and Matthieu Cord. Cross-modal retrieval in the cooking context: Learning semantic text-image embeddings. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 35–44, 2018.

Jing-jing Chen, Chong-Wah Ngo, and Tat-Seng Chua. Cross-modal recipe retrieval with rich food attributes. In *Proceedings of the 25th ACM international conference on Multimedia*, pages 1771–1779, 2017a.

Jing-jing Chen, Chong-Wah Ngo, and Tat-Seng Chua. Cross-modal recipe retrieval with rich food attributes. MM '17, page 1771–1779, New York, NY, USA, 2017b. Association for Computing Machinery. ISBN 9781450349062. doi: 10.1145/3123266.3123428. URL `https://doi.org/10.1145/3123266.3123428`.

Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan Routledge, and William Yang Wang. FinQA: A dataset of numerical reasoning over financial data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, November 2021.

Jason Choi, Saar Kuzi, Nikhita Vedula, Jie Zhao, Giuseppe Castellucci, Marcus Collins, Shervin Malmasi, Oleg Rokhlenko, and Eugene Agichtein. Wizard of tasks: A novel conversational dataset for solving real-world tasks in conversational settings. In *COLING 2022*, 2022. URL `https://www.amazon.science/publications/wizard-of-tasks-a-novel-conversational-dataset-for-solving-real-world-tasks-in-conversational-settings`.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*, 2022.

Amazon Alexa Cobot. Alexa taskbot cobot toolkit bert based response selector. Retrieved from `https://mainline.d1rg0zke5cd0dg.amplifyapp.com/doc/htm/bert_response_selector.html`, 2023.

Matthias Egli. Instafoodroberta-ner. `https://huggingface.co/Dizex/InstaFoodRoBERTa-NER`, 2023. Accessed: 2023-07-22.

Bahare Fatemi, Quentin Duval, Rohit Girdhar, Michal Drozdzal, and Adriana Romero-Soriano. Learning to substitute ingredients in recipes. *arXiv preprint arXiv:2302.07960*, 2023.

Peter C Gordon, Randall Hendrick, and Marcus Johnson. Effects of noun phrase type on sentence complexity. *Journal of memory and Language*, 51(1):97–114, 2004.

Paul Green and Lisa Wei-Haas. The rapid development of user interfaces: Experience with the wizard of oz method. In *Proceedings of the human factors society annual meeting*, volume 29, pages 470–474. SAGE Publications Sage CA: Los Angeles, CA, 1985.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. spaCy: Industrial-strength Natural Language Processing in Python. 2020. doi: 10.5281/zenodo.1212303.

Siying Hu, Hen Chen Yen, Ziwei Yu, Mingjian Zhao, Katie Seaborn, and Can Liu. Wizundry: A cooperative wizard of oz platform for simulating future speech-based interfaces with multiple wizards. *Proceedings of the ACM on Human-Computer Interaction*, 7(CSCW1): 1–34, 2023.

Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. Unifiedqa: Crossing format boundaries with a single qa system. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1896–1907, 2020.

Daniel Khashabi, Yeganeh Kordi, and Hannaneh Hajishirzi. Unifiedqa-v2: Stronger generalization via broader cross-format training. *arXiv:2202.12359*, 2022.

Chandra Khatri, Behnam Hedayatnia, Anu Venkatesh, Jeff Nunn, Yi Pan, Qing Liu, Han Song, Anna Gottardi, Sanjeev Kwatra, Sanju Pancholi, et al. Advancing the state of the art in open domain dialog systems through the alexa prize. *arXiv preprint arXiv:1812.10757*, 2018.

Thomas Lin, Oren Etzioni, et al. No noun phrase left behind: detecting and typing unlinkable entities. In *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*, pages 893–903, 2012.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Edward Loper and Steven Bird. Nltk: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics*, pages 63–70, 2002.

Yujie Lu, Pan Lu, Zhiyu Chen, Wanrong Zhu, Xin Eric Wang, and William Yang Wang. Multimodal procedural planning via dual text-image prompting. *arXiv preprint arXiv:2305.01795*, 2023.

Amaia Salvador, Nicholas Hynes, Yusuf Aytar, Javier Marin, Ferda Ofli, Ingmar Weber, and Antonio Torralba. Learning cross-modal embeddings for cooking recipes and food images. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3068–3076, 2017. doi: 10.1109/CVPR.2017.327.

Mustafa Shukor, Guillaume Couairon, Asya Grechka, and Matthieu Cord. Transformer decoders with multimodal regularization for cross-modal food retrieval. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4567–4578, 2022.

Hao Wang, Doyen Sahoo, Chenghao Liu, Ee-peng Lim, and Steven CH Hoi. Learning cross-modal embeddings with adversarial networks for cooking recipes and food images. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11572–11581, 2019.

Bin Zhu, Chong-Wah Ngo, Jingjing Chen, and Yanbin Hao. R2gan: Cross-modal recipe retrieval with generative adversarial network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.