EvoquerBot: A Multimedia Chatbot Leveraging Synthetic Data for Cross-Domain Assistance

Ranran Haoran Zhang , Parker Sell , Yusen Zhang , Liwei Che , Allen Gao
Sathiyajith K S , Rishabh Bhatt , Piyush Nagasubramaniam , Sujeeth Vummanthala
Salika Dave , Harsh Maniar , Vishnu Dasu , Rui Zhang

Penn State University

`{hzz5361, pqs5479, yfz5488, lfc5481, jkg5673, sxk6394, rishabh.bhatt, pvn5119, srv5245, sad6003, hmx5250, vdasu, rmz5227}@psu.edu`

Abstract

EvoquerBot, developed for the TaskBot challenge, is a multimedia chatbot designed to assist users in completing Cooking and DIY tasks within a single session. The bot leverages a coordinated orchestration of submodules for intent classification, task recommendation, task description, and step navigation. This paper addresses the challenges of short development and model training time, data quality in both NLP and multimedia sectors, multimedia response handling, and tailoring the conversation flow to domain-specific user experiences. To overcome these, we propose agile classifier development, data augmentation, multimedia response design, and domain-specific dialogue state machines. The conversation flow is governed by an efficient intent classifier and a recursion-based state machine, further enhanced with features such as Cooking Image Augmentation and DIY Substep Decomposition. The effectiveness of our system is validated by the superior relevance of task recommendations, demonstrating its ability to enhance user experience.

1 Introduction

This paper introduces **EvoquerBot**, a multimedia chatbot developed as part of the *TaskBot* challenge (Agichtein et al., 2023). This challenge necessitates the creation of conversational agents capable of assisting humans with real-world tasks, employing both voice and visual modalities. EvoquerBot is specifically designed to guide users in the Cooking and DIY domains to complete tasks within a single session, leveraging a well-coordinated orchestration of submodules for intent classification, task recommendation, task description, and step navigation.

We address several challenges in this field: (1) the short development and model training time between each system iteration (Sculley et al., 2015; Siebert et al., 2020; Zhang et al., 2022b), (2) the lack of high-quality data in both NLP and multimedia sectors, specifically the hard-to-predict voice commands from the real world (Ham et al., 2020; Peng et al., 2020b; Zhang et al., 2021; Xia et al., 2021; Zhang et al., 2022a), (3) handling multimedia responses in terms of both input and output (Furui, 1995; Nie et al., 2019; Cui et al., 2019; Wang et al., 2023), and (4) tailoring the conversation flow to domain-specific user experiences (Li et al., 2022; Pan et al., 2022). To tackle these challenges, we propose the following solutions:

- **Agile Classifier Development**: The ever-evolving user feedback necessitated an adaptive approach toward classifier development. This ensures that EvoquerBot evolves in parallel with user needs, maintaining relevance and utility.

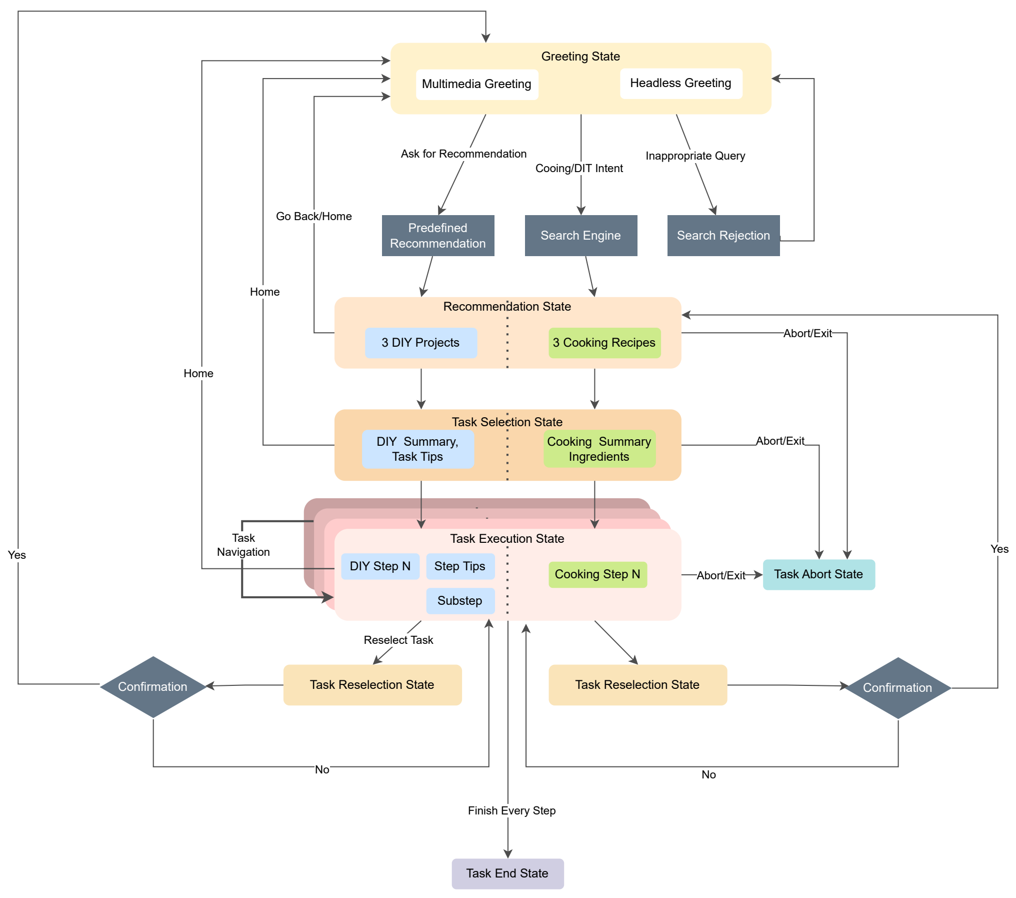


Figure 1: The overview of the state machine

- **Data augmentation**: we employ GPT4 and CLIP for NLP and multimodal offline data augmentation, and also design effective definite algorithms for online system.
- **Multimedia Response Design**: We meticulously craft multimedia responses to maximize the chances of success in the next turn of the conversation and provide appropriate guidance to first-time users.
- **Domain-Specific Dialogue State Machines**: Recognizing the distinct user expectations within the Cooking and DIY domains, we developed separate dialogue state machines to cater to each domain appropriately.

2 System Overview

Our system is integrated with the cobot framework (Khatri et al., 2018). In our setup, the cobot is tasked with key functions such as Automatic Speech Recognition (ASR) and User Data Management, which is handled in DynamoDB. It also manages the input and output formatting for the Alexa Presentation Language, which includes touch screen input and multimedia screen formatting. The core functionalities of EvoquerBot resides in a remote module, hosted on auto-scaled EC2 clusters, ensuring low latency and scalability in response to dynamic user traffic.

Communication between the CoBot and the remote module is facilitated via a Flask web service. This service securely stores all user inputs, bot outputs, and metadata in a "session" JSON object. For instance, when a new user interacts with the system, an empty session, along with the first user utterance, is initialized and sent from the CoBot to the remote module. The session is then processed, updated with the bot response, and enriched with user and task-specific metadata before being sent back to the CoBot. In subsequent interactions, the CoBot retrieves the updated session from DynamoDB and sends it to the remote module.

The following section primarily discusses the EvoquerBot within the remote module. The conversation flow is governed by an efficient intent classifier (section 3.2) and a recursion-based state machine. We further enhance the conversation flow with functionalities such as Cooking Image Augmentation (section 4.4), step time duration prediction (section 4.2), and DIY Substep Decomposition (section 4.3).

3 Core Dialogue Logics

3.1 State Machine

As illustrated in Figure 1, we implement distinct logic for multimedia and headless devices within each state of the state machine. To further enhance the user experience, we customize the intent flows from the recommendation state for both the DIY and cooking domains, reflecting the different user behaviors observed within these areas.

For instance, when users wish to re-select a task within the cooking domain, their intent typically involves slight modifications to the original search query due to constraints such as missing ingredients or dietary restrictions. In response, our system directs them back to the recommendation state, presenting a set of similar recipes for their consideration. This allows users to easily refine their initial query with more specific details, like the desired type or color of a cake, and receive a refreshed list of recommended recipes.

Conversely, within the DIY domain, users who decide to re-select a task usually aim to switch tasks entirely. Suppose a user initially chose a DIY task to build a bookshelf but then opts for a different project, such as crafting a birdhouse. In this scenario, our system reverts to the greeting state and prompts for a new query.

Our state machine's design incorporates these behavioral nuances, ensuring that the task re-selection process caters effectively to user needs across both domains. Given the current state and the user utterance, the next state and the corresponding predefined response will be determined by intent classification and the state machine.

3.2 Intent Classification

Our initial strategy for intent classification employed a GPT-2-based model (Radford et al., 2019), trained on the open-source Wizard-of-Tasks dataset. Although GPT-family models theoretically have the capability to concurrently generate intent, slot, and response in a single inference, existing research predominantly focuses on few-shot response generation (Peng et al., 2020a). The exploration of integrated intent classification and dialogue response generation in resource-constrained settings remains relatively under-investigated (Hosseini-Asl et al., 2020). Upon regression testing with last year's conversation logs, this approach exhibited a significant shortfall, with accuracy rates plummeting below 40%. In light of the continually evolving user feedback and obligatory certification checks, the utility of a purely neural-based intent classifier is increasingly being called into question. As such, a purely neural-based intent classifier was deemed impractical.

For instance, queries such as "Is Siri better than Alexa" fall under safety checks, classified as sensitive intents, and consequently, EvoquerBot abstains from providing a response. Feedback from users provided another example, as they often expressed "complete" or "done" after finishing a step, indicating a "state end" intent The integration of this new intent into our existing classifier resulted in conflicts with the "task end" intent.

Due to these challenges, we ultimately phased out the neural classifier in favor of manual adjustments to the keyword list, ensuring mutual exclusivity. Our finalized list of intents, along with their optional arguments, is outlined in Table. 1.

The keyword-based intent algorithm we used originated from Grillbot of Glasgow (2022). We addressed a performance bug and the updated version is outlined in Algorithm 1. This algorithm has a time complexity of $O(mn)$, where m represents the token length of the user utterance and n is the length of the keyword list. To enhance the algorithm's efficiency, we employed the Aho-Corasick

Algorithm 1: Algorithm for the Keyword Classifier. Time Complexity $O(mn)$.

```
Data: Path to the wordlist: wordlist_path, User utterance: user_utterance
Result: A Boolean: True if any words or phrases from the wordlist are found in the user
        utterance, else False
Initialization;
Open the file at wordlist_path and read in all lines to a list, all_word_or_sentence_list, with each
  line stripped of whitespace and lowercased;
Process the user_utterance;
Set user_utterance to lowercase;
Tokenize user_utterance into a list, tokenized_user_utterance using NLP model;
for each word_or_sentence in all_word_or_sentence_list do
    Split word_or_sentence into a list, words;
    if length of words > 1 then
        Assign word_or_sentence to sentence;
        if sentence is found in user_utterance then
            return True
        end
    else
        Assign word_or_sentence to word;
        if word is found in tokenized_user_utterance then
            return True
        end
    end
end
return False
```

Algorithm 2: Improved Algorithm for the Keyword Classifier using Aho–Corasick (AC) Automation. AC Automation is a beneficial data structure for string searching, providing an improvement from $O(mn)$ to $O(n + m + z)$, where z is the output size.

```
Data: Path to the wordlist: wordlist_path, User utterance: user_utterance
Result: A Boolean: True if any words or phrases from the wordlist are found in the user
        utterance, else False
Initialization;
Open the file at wordlist_path and read in all lines to a list, all_word_or_sentence_list, with each
  line stripped of whitespace and lowercased;
Create a new automaton, ac, using Aho–Corasick algorithm;
for each word_or_sentence in all_word_or_sentence_list do
    Create a token from word_or_sentence with surrounding $ symbols and $ symbols between
      words;
    Add the token to the ac automaton with it's index and itself as data;
end
Make the ac automaton;
Classification;
Process the user_utterance; Set user_utterance to lowercase;
Create a modified version of user_utterance, mod_user_utterance, surrounded by $ symbols and $
  symbols between words;
for each item in the longest items in ac that match mod_user_utterance do
    Print the item;
    Set the return boolean, ret, to True;
    return ret;
end
return ret;
```

(AC) Automation Aho and Corasick (1975) (refer to Algorithm 2). This method utilizes a data structure optimized for string search, resulting in a complexity of $O(n + m + z)$, where z denotes the output size.

Algorithm 3: Task Selection Intent Classification.

```
Data: input_dict : SessionData
Result: Modified input_dict
begin
    // Initialize the list of lower-cased task titles
    title_list ← LowerCaseTitles(input_dict.task_candidates)
    // Check if there are tasks and if a relevant recommendation was
       previously made
    if TaskRecommendationExist then
        // Transform touch screen event to text
        input_dict.user_input ← input_dict.touchscreen
        // Preprocess the user's input
        text ← LowerCase(input_dict.user_input)
        doc ← TokenizeText(text)
        // Attempt task selection based on ordinal or numeric indication.
           e.g., one, two ...
        AttemptTaskSelectionByOrdinals(doc, input_dict, SelectionQuery.to_list())
        // Attempt task selection based on 'option' keyword context.  e.g.,
           Option 1, 2 ...
        AttemptTaskSelectionByOptionKeyword(doc, input_dict, 'option', index2str)
        // Attempt task selection based on noun phrase matching
        AttemptTaskSelectionByNounPhraseMatching(doc, title_list, input_dict, process,
          index2str, matching_score_threshold=75)
    else
        return input_dict;
    end
end
```

Ensemble Recommendation Classifier: The classification of the recommendation intent is executed via an ensemble approach that integrates the Alexa Prize CoBot API (domain classifier), a GPT2 Classifier, and a Keyword Classifier. Splitting the two domains, Cooking and DIY, in this process enhances the user experience by providing more targeted tasks and conversation flow. For instance, we discovered some ambiguous recipes in the Wikihow dataset, like “how to cook meat in Minecraft”. Thus, by segregating the domains, we can ensure that the system is more likely to recommend real-world cooking tasks when in the cooking domain, and similarly, actual DIY tasks when in the DIY domain. This distinction mitigates the risk of recommending tasks that are contextually incorrect or irrelevant.

Task Selection: On multimedia devices, users have different ways to initiate a task. They may either say “Option 2” to select from a list of tasks or directly voice the task title. In some instances, users may seek new tasks and voice a new search query directly. Distinguishing between the latter two scenarios can be challenging, given their close nature. Nevertheless, our system is designed to accurately interpret user intent in all these scenarios. The full algorithm is demonstrated in Algorithm. 3.

3.3 User Response

To enhance the conversational experience for users, we leverage the capabilities of GPT-4 (OpenAI, 2023), the latest large-scale language model from OpenAI. We harness its generative capabilities to create an extensive set of utterance templates that serve as potential responses to user queries.

In the context of task-based conversational agents, the capacity to provide varied responses is crucial in maintaining user engagement and fostering a more natural dialogue flow. Our approach addresses this need. We feed user queries to GPT-4, instructing it to generate a collection of possible responses.

Intent	Optional Argument
Safety Check	Financial, Medical, Sensitive ...
Recommendation	Themed event, Cooking, DIY
Navigation	Next, Previous, Repeat, Last Step
Confirmation	Yes, No
State End	Next Step, Task End
Task Selection	Option N, Title Fuzzy Match ...
Task End	Abortion, Complete
Alternate Task	-
Substep	-
Tips	-
Ingredients	-

Table 1: Classification of User Intents in EvoquerBot with Optional Arguments.

These generated utterances encompass a broad spectrum of phrasing and semantic nuances, thereby accommodating diverse conversational styles and user preferences.

To prevent repetitive or monotonous conversation flows, we employ a random selection process when choosing from these generated templates. This introduces an element of variation into our response generation, ensuring that users experience a diverse and dynamic range of responses during their interactions with EvoquerBot.

By combining the generative power of GPT-4 with a randomized selection strategy, we succeed in fostering a more engaging, diverse, and organic conversational experience for users.

4 Functionalities

4.1 Recommendation Engine

	Lucene	Lucene + Fuzzy Reordering
DIY Query	alexa show me to make origami	
Top 3 Tasks	How to Seduce Your Girlfriend How to Make an Origami Wallet How to Choose Paper for Origami	How to Make an Origami Heart How to Make an Origami Wallet How to Choose Paper for Origami
DIY Query	how to paint a concrete wall	
Top 3 Tasks	How to Paint a Concrete Wall How to Remove Paint from a Concrete Porch How to Stop Efflorescence	How to Paint a Concrete Wall How to Paint Concrete How to Remove Paint from a Concrete Porch
Cooking Query	how to cook nachos?	
Top 3 Tasks	Turkey Nachos with Cranberry Salsa Black-Eyed Pea Nachos Hearty Rice and Bean Vegetarian Nachos	Bay Scallop Nachos with Black Beans and Corn Black-Eyed Pea Nachos Hearty Rice and Bean Vegetarian Nachos
Cooking Query	Recipe for apple pie	
Top 3 Tasks	Rustic Apple Galette Pumpkin Apple Pie Sweet Potato Apple Pie	Cinnamon-Apple Slab Pie Apple Pie Smoothie Green Apple Syrup

Table 2: Comparison of top 3 task recommendations using original Lucene indexing and our multi-step (Lucene + Fuzzy Reordering) recommendation engine.

We employed the pyserini package mentioned in of Glasgow (2022) to create the offline recommendation index. The structured task data, provided by Amazon, were sourced from WikiHow for DIY tasks and Whole Foods Market for cooking tasks. To create this index, we utilized a Bag-of-Words (BOW) based Lucene Indexing, treating the Cooking and DIY domains separately due to their unique characteristics. The online recommendation system, on the other hand, operates through a straightforward three-step process.

Upon receiving a user query, such as “show me how to make origami”, the system instigates a three-tier recommendation process. (1) A domain classifier, powered by the Alexa Prize CoBot

API, sorts the query into the appropriate domain, be it Cooking or DIY. Subsequently, the query is processed by our information retrieval engine, which compiles a list of the top 20 tasks (derived from the entire task paragraph) that exhibit the greatest BOW similarity to the user's request. (2) The process proceeds to a critical filtering stage. Here, a keyword-based classifier scrutinizes the initial list of retrieved tasks and dismisses any tasks considered unsuitable or irrelevant for the given context. For instance, tasks such as "How to Apply for a Concealed Carry Permit" are deemed inappropriate for family settings and thus filtered out. (3) The final step involves title-based fuzzy reordering. We discovered that the results returned by the second step often fail to prioritize the most relevant titles at the forefront, due to the task-paragraph based information engine favoring recall over title relevance. As a solution, we further employ title-based string similarity reordering (utilizing the fuzzywuzzy Python library's fuzz.partial_token_sort_ratio function) to present the top 3 tasks to the users, ensuring they are the best matches.

Table 2 showcases the superior relevance of task recommendations achieved by our multi-step process in contrast to the original Lucene-based approach. For example, the Lucene recommendation for an origami-related query included unrelated tasks like "How to Seduce Your Girlfriend". However, our system accurately suggested tasks directly associated with origami. This degree of accuracy and relevance is uniformly observed across the DIY and cooking domains, validating the effectiveness of our system in supplying appropriate task recommendations and enhancing the user experience.

4.2 Time Duration Prediction

Time duration prediction aims to predict the duration of each step in each task. We introduce two types of duration: Operation Time, serving as the time budget for the user to operate the current step; and Idle Time, representing the time that a user needs to wait after doing the operation. This functionality is a key step for dynamic routing. Specifically, users can leverage the Idle Time to move on to the next step. For instance, during "preheat the oven to 450 degrees", users can dynamically move on to the next step.

Time Duration Predictor: We simply apply ChatGPT API to predict time duration. The input to the model consists of instructions for predicting time duration and the text of one step to the model. To evaluate the quality of the results, we manually label 10 tasks with 64 steps. A true label is assigned if the result is acceptable in real-world scenarios and vice versa. We obtain an 87.6% accuracy on the test samples showing the quality of the predicted results.

Recipe	Step	Predicted Time	Label
Fennel and Kale Salad with White Beans and Mint	In a large bowl, whisk together mayo, vinegar, parsley, salt and pepper.	Operation Time: 2 minutes Waiting (idle) Time: 0 minutes	True
Fennel and Kale Salad with White Beans and Mint	Add mint, fennel, kale and beans, gently toss together and serve immediately.	Operation Time: 2 minutes Waiting (idle) Time: 0 minutes	True
Homemade Fennel Mustard	Stir in more water if it becomes too thick.	Operation Time: 1 minute Waiting (idle) Time: 0 minutes	False
Homemade Fennel Mustard	The mustard will keep refrigerated for several months.	Operation Time: 0 minutes Waiting (idle) Time: 0 minutes	False
Vegan Deviled "Eggs"	Preheat the oven to 350°F. Line a baking sheet with parchment paper.	Operation Time: 5 minutes Waiting (idle) Time: 0 minutes	False
Vegan Deviled "Eggs"	Cut each potato in half crosswise.	Operation Time: 2 minutes Waiting (idle) Time: 0 minutes	False

Table 3: Examples of the time predicted results on recipe dataset.

Rule-based Corrector: As shown in Table 3, two types of errors are made frequently by the model. The first type is to predict zero minutes. We use the rule to filter out all zero-minute results and replace them with 0.5 minutes to increase the interpretability for the users. Second, the preheating step always has the wrong results, we also filter out all steps with the keyword "preheat" and set the Idle Time to 10 minutes.

4.3 Substep segmentation

The length of the step varies in our datasets, however, when Alexa read one step, the attention of the users is only around 20 seconds. If the step is too long, the user may not be able to follow the instruction which leads to inconvenience. Thus, we split the long step into smaller units, so-called substeps. During the job, the Alexa will read one substep and pause, then move on to the next substep to make the user easier to track.

Rule-based Sgementor: We use a simple but effective rule to split the long step text (Algorithm. 4). First, we use NLTK[1] to conduct sentence tokenization. And then use it to do pos tagging for the first token in the sentence. Then, we check the first token of each sentence, if it is a verb or gerund, we mark it with "B" which means the beginning of one substep. Otherwise, it is marked as "I" which is the intermediate sentence of one substep. Next, we concatenate a "B" sentence with all following "I" sentences until meeting the other "B" or end of the step. For instance, if one step is marked as "BIBIIB", it will be split into three substeps, with two, three, and one sentences, respectively. Compared with the ChatGPT based methods, this simple approach enjoys high accuracy without any information loss and lower time/money cost.

Algorithm 4: Rule-based Substep Segmentation Algorithm.

```
Data: Text of a step $S$
Result: A list of substeps $L$. Each element represents a substep.
Initialization;
Initialize a substep list $L = \Phi$;
Use NLTK to split $S$ into $k$ sentences: $s_1, s_2, \cdots, s_k$;
Initialize $cur\_substep = \Phi$;
for each $i \in [1, k]$ do
    Use NLTK to tokenize $s_i$ into words $w_1, w_2, \cdots, w_l$;
    Use NLTK to get the pos tagging of $w_1$, if pos tagging is a verb or gerund, assign this
      sentence a "B", otherwise, assign this sentence a "I";
    if Assigned label is "B" then
        Append $cur\_substep$ to $L$ if it is not empty, $cur\_substep = \Phi$;
    end
    Append $s_i$ to $cur\_substep$;
end
if $cur\_substep$ is not empty then
    Append $cur\_substep$ to $L$;
end
return $L$;
```

4.4 Image Augmentation

The combination of text description and image display can provide users with an intuitive and specific task guidance experience. However, the majority of the DIY tasks and recipes contain step texts only, lacking step-wise pairing images for better instruction understanding. To resolve this challenge and provide multimodal information for the users, the EvoquerBot is equipped with image augmentation functionality to match a pairing image for each steps in the tasks.

The image augmentation functionality aids the user through visual cues by providing a relevant image for each step across all the tasks. The functionality consists of two core parts, the cross-modal retrieval module and the text-to-image generation module. The former matches an appropriate image for each text step. The latter supplements text that lacks suitable images. The advanced image synthetic model is used for prompt-augmented text-to-image generation. We introduce further detail in the following subsections with a focus on image augmentation for recipes.

[1] https://www.nltk.org/

4.4.1 Data Collection

The effectiveness of this image augmentation strategy is heavily dependent on the quality of the data collected. Taking the matching of recipe steps with images as an example, on one hand, matching each step with an appropriate image requires a sufficient amount of relevant images. On the other hand, evaluating the quality of a step-image pair is subjective except for cases in which the retrieved image is completely irrelevant to the step. Thus, it is a non-trivial task to evaluate the quality of the proposed strategy at scale.

In order to minimize egregious step-image mismatches in the following retrieval step, we curated images from recipes based on different features to make the dataset as diverse as possible. These features include cooking time (30-minute meals, quick meals, etc.), number of ingredients (meals made using 5 ingredients or less), primary ingredient (meat and poultry, vegetarian, fish), healthy dishes, geographical region (American cuisine, Asian cuisine, and more), etc. Our data collection strategy aims to provide effective visual cues irrespective of the type of cuisine, geographical region, or other characteristics.

The image scraping was done using *beautifulsoup* and *selenium*. In addition to the image URLs, the alt-text and captions were also scraped. The necessary images were scraped from popular websites including seriouseats and allrecipes. A total of $29,000$ images were chosen for the cooking domain.

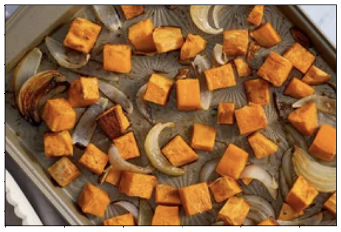
4.4.2 Cross-modal Retrieval

The cross-modal retrieval process leverages the zero-shot capabilities of the CLIP model Radford et al. (2021). Typical zero-shot use cases of CLIP accept multiple text inputs and a single image input to find the best image-text pair. However, this approach is not efficient when trying to generate step-image pairs for all steps in the taskgraph dataset, due to the exponential combination and computation cost.

In order to reduce the computation cost when transforming original data into embeddings via CLIP encoders, we precompute the image embeddings for all the scraped images and the text embeddings for all the task step sentences. The step-image scores are determined by computing the cosine similarity between the precomputed embeddings. In such a case, the CLIP embeddings are not recomputed multiple times, and the CLIP model is no longer needed in memory after computing the embeddings. The step-image alignment algorithm computes the similarity scores by comparing batches of text embeddings against each image for faster execution time and better GPU utilization.

The above-mentioned retrieval process can be further enhanced by using the ingredient list of recipes, which is stored as metadata in the taskgraphs. Using a two-step approach, we first map a subset of candidate images to each recipe/taskgraph by computing the retrieval score of the ingredients and the images. In this case, the text embeddings of the ingredient lists need to be precomputed as well. In the second step, we find a matching image for each step based on the subset of images mapped to that recipe instead of searching all the images. The ingredient lists allow us to establish some context for the recipe and the search strategy. The step texts may have higher scores with irrelevant images due to utensils or actions in shown in the image. Using this approach, irrelevant images are filtered based on the ingredients of the recipe.

We demonstrate our results of the cross-modal retrieval strategy in Table 4.

Text	Image
Cut each sweet potato lengthwise into 1/2 inch strips then arrange on the same foil-lined baking sheet in a single layer.	

Continued on next page

Table 4: Examples of Step-Image pairs using CLIP-based cross-modal retrieval

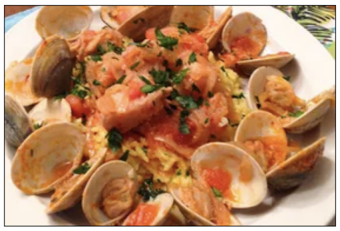
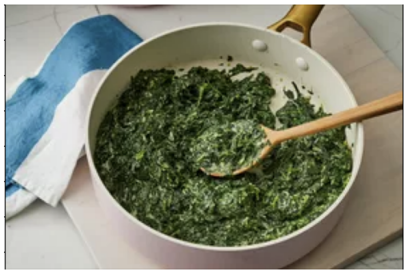
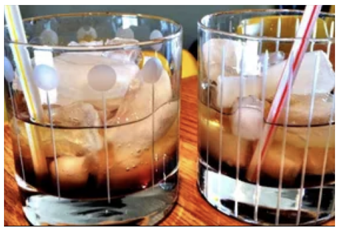
To serve, spoon clams and chorizo over salmon and garnish with additional parsley.	
Add spinach, watercress, and lettuce and toss gently to coat.	
Place ice cubes, elderflower liqueur and bitters in a glass on the rocks	

Table 4: Examples of Step-Image pairs using CLIP-based cross-modal retrieval (Continued)

4.4.3 Text-to-Image Models for Generating Synthetic Data

Although the data collection strategy creates a sufficiently diverse dataset, there will invariably be bad step-image pairs. The cases in which the cross-modal retrieval score is below an acceptable threshold are addressed using text-to-image generative models. There are around 6% step texts having difficulty finding suitable images with high matching scores. Most of them describe the preparation stages and steps with complex operations. To address this issue, we consider adapting advanced text-to-image generation models such as DALL-E2, which has shown promising results for generating images related to cooking tasks. However, the quality of image generation in terms of relevance and realism is very sensitive to the input text prompt.

In order to generate suitable images for the user, we use a combination of keyword extraction and special modifiers to craft effective prompts. An overview of the rule-based prompt engineering process is given below.

Keyword Extraction: Text prompts that are too long, too short, or contain redundant information tend to generate images that are irrelevant to the step or unrealistic in appearance. Instead of using the raw step text as input for image generation, we extract the most relevant keywords using the RAKE algorithm Rose et al. (2010). The keywords are concatenated to create a part of the prompt.

Modifiers: The images generated should not only be relevant but also be visually appealing. Incomplete or unspecific prompts can generate images that are unpleasant to the eye. For example, images of utensils that appear to be unhygienic are likely to be off-putting to users, especially while cooking. Good modifiers can help maintain image quality and relevance during large-scale image generation. These modifiers include “An aesthetic picture of X”, “X appetizing”, and “in kitchen”. “X” refers to the concatenated text obtained from the keyword extraction step. With regard to image relevance, it is challenging to generate images for steps that show cooking in progress and when the dish is yet to be completed. Without using modifiers, DALL-E2 tends to generate images of either cooked dishes or raw ingredients. Appending the modifier "being" to the prompt helps generate images relevant to "in-progress" cooking steps.

Table 5 presents a comparison of the results obtained from using the original text and the augmented text, which is based on the prompt engineering pipeline. In the case of the kitchen pan, the original prompt, "Oil a large pan and set aside.", generates a rather unsightly image while the augmented prompt generates a much more suitable image in the context of cooking. The third example presented in Table 5 shows how our prompt engineering technique can craft a prompt to generate "in-progress" cooking images, unlike the original prompt. It is worth noting that the augmented prompt lacks semantic clarity compared to the original prompt because the order of the words is jumbled and in other cases, the prompt might be truncated. Despite the difference in semantic clarity, the modifiers and chosen keywords help craft a prompt that can generate images relevant to the cooking domain better than the original prompt.

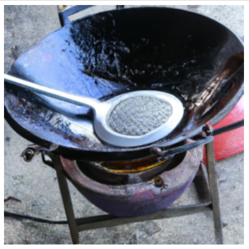
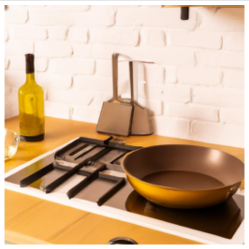
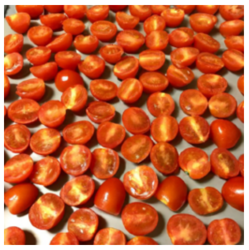
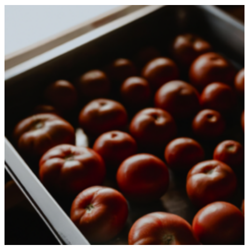
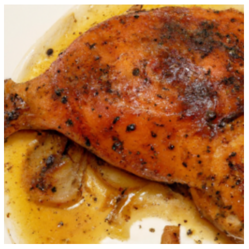
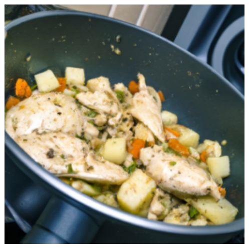
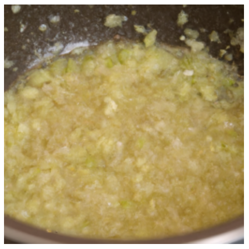
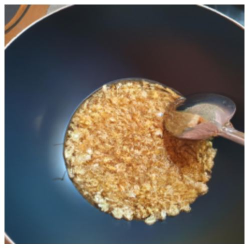
Original Step Text	Image	Augmented Prompt	Image
Oil a large pan and set aside.		Aesthetic picture of set aside large pan oil in kitchen	
Arrange tomatoes on a large rimmed baking sheet in a single layer.		Aesthetic picture of large rimmed baking sheet single layer arrange tomatoes in kitchen	
Roast, basting occasionally with pan juices, until chicken is cooked through and vegetables are very tender, about 45 minutes.		Aesthetic picture of pan juices basting occasionally 45 minutes vegetables tender roast cooked chicken in kitchen	
Add garlic and cook, stirring frequently, until the garlic starts to brown, about 1 minute.		Aesthetic picture of stirring frequently garlic starts add garlic 1 minute cook brown in kitchen	

Table 5: Examples showcasing the results of Prompt Engineering Inputs for DALL-E2 (Continued)

4.4.4 Future Work

The CLIP model has been particularly useful for cross-modal retrieval, but it is worth exploring how the alt-text and image captions can be used to improve the quality of the step-image alignment algorithm. The scraped metadata is often not descriptive enough and lacks relevance to the corresponding cooking steps, making it challenging to use semantic similarity as a metric for evaluating step-image matches.

DALL-E2 has shown promising results in generating synthetic data when the scraped images are lacking. It is challenging to evaluate the quality of a prompt-engineering strategy because the image generation process in DALL-E2 is stochastic in nature i.e. the same prompt can result in different images being generated.

5 User Data Analysis

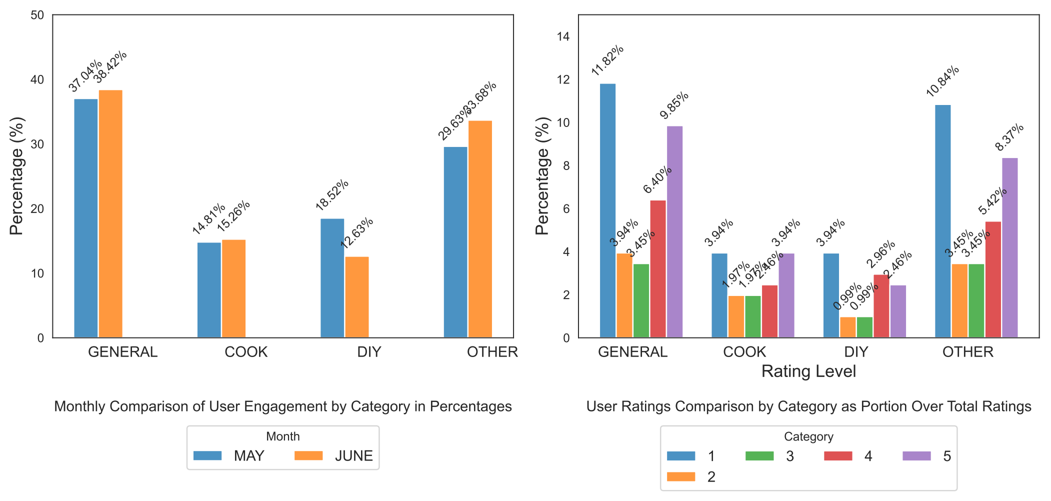


Figure 2: Evoquer User Statistics.

For our analysis, we used data collected in May and June through a real-time system designed to scrape and monitor user transcripts, scores, and feedback. As depicted in Figure 2, this system allowed us to gather critical insights into user interactions and performance metrics during these two months. The left image demonstrates the distribution of conversations across various categories, offering a snapshot of user engagement with different tasks. The right image further dissects the conversation count by user score within each category, providing a more nuanced understanding of user satisfaction and task completion rates. Importantly, this real-time monitoring system served as a dynamic tool for refining EvoquerBot's performance. It allowed us to update our keyword list in response to user interactions and to remove tasks that were deemed inappropriate or challenging to find.

6 Conclusion

In this work, we present **EvoquerBot**, a multimedia chatbot that integrates advanced AI models to provide a comprehensive and interactive user experience. The system effectively scrapes images from popular websites, aligns them with corresponding text steps, and leverages the CLIP model for efficient cross-modal retrieval. Despite challenges with generating relevant images from incomplete or unspecific prompts, EvoquerBot has shown its potential in crafting effective prompts for image generation using models like DALL-E2. The bot also employs a sophisticated conversation flow management system, governed by an efficient intent classifier and a recursion-based state machine, to cater to user needs across both cooking and DIY domains. As we look to the future, we see promising avenues for exploration and improvement, such as the potential use of alt-text and image captions to enhance the step-image alignment algorithm.

References

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. 2023. Alexa, let's work together: Introducing the second alexa prize taskbot challenge. In *Alexa Prize TaskBot Challenge 2 Proceedings*.

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

Chen Cui, Wenjie Wang, Xuemeng Song, Minlie Huang, Xin-Shun Xu, and Liqiang Nie. 2019. User attention-guided multimodal dialog systems. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR'19, page 445–454, New York, NY, USA. Association for Computing Machinery.

Sadaoki Furui. 1995. Prospects for spoken dialogue systems in a multimedia environment. In *Spoken Dialogue Systems-Theories and Applications*.

University of Glasgow. 2022. Grillbot: A flexible conversational agent for solving complex real-world tasks. In *Alexa Prize TaskBot Challenge Proceedings*.

Donghoon Ham, Jeong-Gwan Lee, Youngsoo Jang, and Kee-Eung Kim. 2020. End-to-end neural pipeline for goal-oriented dialogue systems using GPT-2. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 583–592, Online. Association for Computational Linguistics.

Ehsan Hosseini-Asl, Bryan McCann, Chien-Sheng Wu, Semih Yavuz, and Richard Socher. 2020. A simple language model for task-oriented dialogue. *arXiv preprint arXiv:2005.00796*.

Chandra Khatri, Behnam Hedayatnia, Anu Venkatesh, Jeff Nunn, Yi Pan, Qing Liu, Han Song, Anna Gottardi, Sanjeev Kwatra, Sanju Pancholi, Ming Cheng, Qinglang Chen, Lauren Stubel, Karthik Gopalakrishnan, Kate Bland, Raefer Gabriel, Arindam Mandal, Dilek Hakkani-Tür, Gene Hwang, Nate Michel, Eric King, and Rohit Prasad. 2018. Advancing the state of the art in open domain dialog systems through the alexa prize. *CoRR*, abs/1812.10757.

Dingcheng Li, Zheng Chen, Eunah Cho, Jie Hao, Xiaohu Liu, Fan Xing, Chenlei Guo, and Yang Liu. 2022. Overcoming catastrophic forgetting during domain adaptation of seq2seq language generation. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5441–5454, Seattle, United States. Association for Computational Linguistics.

Liqiang Nie, Wenjie Wang, Richang Hong, Meng Wang, and Qi Tian. 2019. Multimodal dialog system: Generating responses via adaptive decoders. In *Proceedings of the 27th ACM International Conference on Multimedia*, MM '19, page 1098–1106, New York, NY, USA. Association for Computing Machinery.

OpenAI. 2023. Gpt-4 technical report.

Yan Pan, Mingyang Ma, Bernhard Pflugfelder, and Georg Groh. 2022. User satisfaction modeling with domain adaptation in task-oriented dialogue systems. In *Proceedings of the 23rd Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 630–636, Edinburgh, UK. Association for Computational Linguistics.

Baolin Peng, Chenguang Zhu, Chunyuan Li, Xiujun Li, Jinchao Li, Michael Zeng, and Jianfeng Gao. 2020a. Few-shot natural language generation for task-oriented dialog. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 172–182, Online. Association for Computational Linguistics.

Baolin Peng, Chenguang Zhu, Michael Zeng, and Jianfeng Gao. 2020b. Data augmentation for spoken language understanding via pretrained language models. *arXiv preprint arXiv:2004.13952*.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Stuart Rose, Dave Engel, Nick Cramer, and Wendy Cowley. 2010. Automatic keyword extraction from individual documents. *Text mining: applications and theory*, pages 1–20.

David Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-Francois Crespo, and Dan Dennison. 2015. Hidden technical debt in machine learning systems. *Advances in neural information processing systems*, 28.

Julien Siebert, Lisa Joeckel, Jens Heidrich, Koji Nakamichi, Kyoko Ohashi, Isao Namba, Rieko Yamamoto, and Mikio Aoyama. 2020. Towards guidelines for assessing qualities of machine learning systems. In *Quality of Information and Communications Technology: 13th International Conference, QUATIC 2020, Faro, Portugal, September 9–11, 2020, Proceedings 13*, pages 17–31. Springer.

Jiangnan Wang, Haisheng Li, Leiquan Wang, and Chunlei Wu. 2023. A multimodal dialogue system for improving user satisfaction via knowledge-enriched response and image recommendation. *Neural Computing and Applications*, 35(18):13187–13206.

Congying Xia, Caiming Xiong, and Philip Yu. 2021. Pseudo siamese network for few-shot intent generation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2005–2009.

Jian-Guo Zhang, Kazuma Hashimoto, Yao Wan, Zhiwei Liu, Ye Liu, Caiming Xiong, and Philip S Yu. 2022a. Are pretrained transformers robust in intent classification? a missing ingredient in evaluation of out-of-scope intent detection. *The 4th Workshop on NLP for Conversational AI, ACL 2022*.

Jianguo Zhang, Trung Bui, Seunghyun Yoon, Xiang Chen, Zhiwei Liu, Congying Xia, Quan Hung Tran, Walter Chang, and Philip Yu. 2021. Few-shot intent detection via contrastive pre-training and fine-tuning. *EMNLP*.

Jie M. Zhang, Mark Harman, Lei Ma, and Yang Liu. 2022b. Machine learning testing: Survey, landscapes and horizons. *IEEE Transactions on Software Engineering*, 48(1):1–36.