Bootstrapping a User-Centered Task-Oriented Dialogue System

Shijie Chen[*], Ziru Chen[*], Xiang Deng[†], Ashley Lewis[†], Lingbo Mo[†], Samuel Stevens[†] Zhen Wang[†], Xiang Yue[†], Tianshu Zhang[†], Yu Su[‡], Huan Sun[‡]

The Ohio State University

{chen.10216, chen.8336, deng.595, lewis.2799, mo.169, stevens.994, wang.9215, yue.149, zhang.11535, su.809, sun.397}@osu.edu

Abstract

We present **TacoBot**, a task-oriented dialogue system built for the inaugural Alexa Prize TaskBot Challenge, which assists users in completing multi-step cooking and home improvement tasks. TacoBot is designed with a user-centered principle and aspires to deliver a collaborative and accessible dialogue experience. Towards that end, it is equipped with accurate language understanding, flexible dialogue management, and engaging response generation. Furthermore, TacoBot is backed by a strong search engine and an automated end-to-end test suite. In bootstrapping the development of TacoBot, we explore a series of data augmentation strategies to train advanced neural language processing models and continuously improve the dialogue experience with collected real conversations. At the end of the semifinals, TacoBot achieved an average rating of $3.55/5.0$.

1 Introduction

This paper presents **TacoBot**, a dialogue system built for the first Alexa Prize TaskBot Challenge, which assists users in completing multi-step cooking and home improvement (DIY) tasks. We envision TacoBot to be *user-centered* and deliver a collaborative and accessible conversational experience. To this end, TacoBot must demonstrate superior capabilities, including accurate language understanding, flexible dialogue management, and engaging response generation.

We face several challenges in achieving the vision: (1) *lack of in-domain training data,* especially given modern neural models are data-hungry and crowdsourcing large-scale annotations is costly, (2) *domain shift of language patterns* from the general domain (on which existing models are trained) to the cooking and DIY domain (the foci of this competition), and (3) *the noisy nature of real user utterances,* which contain long-tail patterns that are difficult for models to understand.

To conquer these challenges, our endeavors include: (1) We explore a series of data augmentation strategies, including leveraging GPT-3 (Brown et al., 2020) to synthesize large-scale training data, which enable us to substantially improve natural language understanding and search and lay a firm foundation for the entire TacoBot system. (2) We annotate real user conversations to build evaluation datasets for most modules. Compared to simulated data, these realistic datasets better reflect the authentic distribution of user inputs and reveal fatal deficiencies of our models. (3) We translate observed user behaviors into actionable design guidelines, based on which we continuously improve

[*]Team Leads. Equal Contribution.

[†]Team members in alphabetical order.

[‡]Faculty advisors.

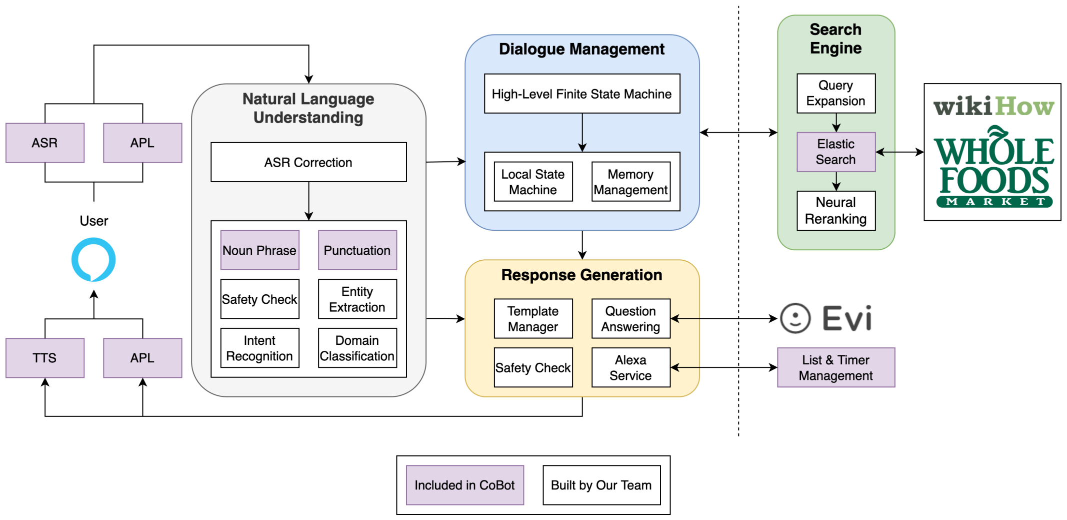


Figure 1: Overall System Design.

our dialogue management module to be more flexible and our generated responses more engaging. (4) We further build an automated end-to-end test suite from scratch, which allows us to identify issues in our system efficiently and fix them before deployment. As a result of all these efforts, among the 528 rated conversations in the semifinals, TacoBot achieved an average rating of $3.55/5.0$ and a task completion rate of 20.08%.

2 System Overview

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

Upon receiving the user input, our system first executes the natural language understanding (NLU) pipeline (Section 3), which first corrects potential ASR errors and then fetches all other modules, hosted as remote EC2 instances. To minimize latency, we use CoBot to run all modules in parallel.

When the NLU pipeline returns results successfully, our system calls the dialogue management (DM) module (Section 4). It relies on a hierarchical finite state machine to control the dialogue flow, handle exceptions, and advance the conversation towards task completion implicitly. Meanwhile, it manages our system's dialogue context in DynamoDB and stores the results of our search engine (Section 5) if detecting a task request. Lastly, DM decides on which response generation modules to execute.

Then, our system runs the selected response generation modules (Section 6) concurrently. Most of the modules are template-based, and we additionally develop a neural question answering module to address user questions. The final textual or multimodal response is rendered by Alexa's SSML Text-To-Speech (TTS) and APL services before delivered to the user.

3 Natural Language Understanding

3.1 TacoBot NLU Pipeline

The NLU pipeline runs at the beginning of each dialogue turn. We blend powerful pre-trained language models and reliable rule-based approaches to build four NLU components: (1) ASR Error Correction, (2) Intent Recognition, (3) Task Name Extraction, and (4) Task Domain Classification.

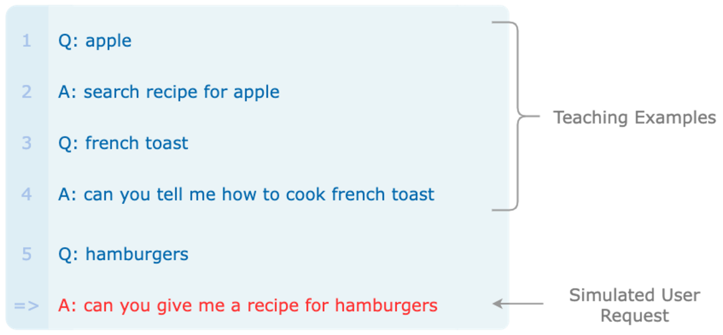
ASR Error Correction. Per our analysis of TacoBot conversations, ASR error is a major cause of user frustration, especially when users try to make a choice or give a command. For example, the

ASR transcription of “step four" often becomes “step for," hindering our system from recognizing this navigation intent. To mitigate this problem, we annotate some common ASR errors in certain dialogue states and correct them by string matching.

Hierarchical Intent Recognition. To support diverse user initiatives, we define 11 dialogue intents under four categories:

- **Sentiment.** On each turn, the user may confirm or reject the bot’s response. Hence, we have three intents, `Affirm`, `Negate` and `Neutral`, to identify the polarity of the user utterance.
- **Commands.** The user can drive the conversation with several commands: **Task Request**, **Navigation** (`More/Less Choice` in Task Catalog to view candidate tasks; `Forward (X Steps)`, `Backward (X Steps)`, and `Go To Step X` in Task Execution to walk through the steps), **Detail Request**, and **Task Complete**. The user can also terminate the ongoing conversation with **Stop** intent at any time.
- **Utilities.** Besides two common utility intents, **Repeat** and **Help**, we have a **Question** intent to capture various user questions throughout the conversation. Also, we support the Alexa list and timer management features using separate intents **List** (`Add` or `Remove` items) and **Timer** (`Set`, `Pause`, `Resume`, or `Cancel` the timer).
- **Exception.** To avoid changing dialogue states by mistake, we have one additional intent for out-of-domain inputs, such as incomplete utterances and greetings.

Moreover, real user initiatives are more complex and may include multiple intents at a time. For instance, *"No, I want to know how to wash my car."* should be classified as both **Sentiment** (`Negate`) and **Task Request** intents. Therefore, we formulate intent recognition as a multi-label classification problem and filter model predictions by dialogue states. Certain intents, such as **Navigation**, are further parsed into fine-grained intents by regular expressions.



```
Q: apple
A: search recipe for apple
Q: french toast
A: can you tell me how to cook french toast
Q: hamburgers
=> A: can you give me a recipe for hamburgers
```

Figure 2: Using GPT-3 to Simulate Task Requests.

To develop this multi-label classification model, we use data augmentation and domain adaptation techniques to get high-quality training data. We first reuse existing datasets (Rastogi et al., 2019) for common conversational intents such as **Sentiment** and **Question**. For other intents, we leverage the in-context learning ability of GPT-3 (Brown et al., 2020), a gigantic pre-trained language model. Specifically, we prompt GPT-3 to synthesize an initial set of utterances with intent descriptions and few-shot examples (Kumar et al., 2020; Yoo et al., 2021), shown in Figure 2. Then, to scale up training data, we transform the synthetic utterances into templates by replacing slot values with placeholders. The templates are filled with various sampled values, such as task names, to get actual training utterances. Moreover, linguistic rules and neural paraphrase models are used to create mixed-intent utterances and improve data diversity. At last, we inject user noise, such as filler words, to enhance the robustness of our intent recognition module.

Task Name Extraction. Given a task request, this module extracts a task name, which will be used by the search engine to construct the search query. As an example, it extracts *“wash a car”* from the home improvement task *“How to wash a car?”* and *“bubble tea”* from the cooking task *“Search bubble tea recipe for me.”* We formulate task name extraction as a span extraction task and fine-tune a BERT-based model (Devlin et al., 2019) on our synthesized task requests.

Task Domain Classification. We train another BERT-based binary classifier on our synthesized task requests to distinguish home improvement and cooking tasks. This enables TacoBot to offer different dialogue experiences for the two different domains.

3.2 Safety Check

TacoBot ensures safe conversations by (1) **Profanity Check**, where we use the offensive speech classifier in CoBot (Khatri et al., 2018a) to check both user utterances and system responses in each turn. For offensive user inputs, our system redirects users to their ongoing tasks. We also remove potentially offensive sentences from the final response if detected. (2) **Task Safety Check.** Following the competition rules, TacoBot politely rejects two kinds of inappropriate task requests: (a) *dangerous tasks*, where users and their properties may get hurt, and (b) *professional tasks*, especially those involving legal, medical, and financial knowledge. To detect these inappropriate task requests, we perform rule-based matching against a keyword blacklist provided by Amazon and a set of over $10,000$ Wikihow article titles annotated by our team. Once detecting (a) *dangerous tasks*, our system directly ends the session; or (b) *professional tasks*, it redirects the user to other appropriate tasks.

4 Dialogue Management

Due to the goal-oriented nature of dialogues in in the Alexa Prize TaskBot challenge, we divide the TaskBot dialogue experience into three phases supported by a utility module and design a hierarchical finite state machine for DM (Figure 3). Each phase has multiple fine-grained dialogue states.

- **Task Search Phase.** Task Search is the first phase in TacoBot, where users try to find a DIY task or recipe. With a specific idea in mind, users can directly issue a query and get search results from our backend search engine. Alternatively, they can ask TacoBot to recommend interesting tasks. Either way, TacoBot can present and compare candidate tasks retrieved by the search engine to help users make a choice. One unique design of TacoBot is that, when searching for recipes, TacoBot actively asks clarification questions regarding diet constraints or cuisine, such as *nuts-free*, *vegetarian*, *Chinese food*, and *Mexican food*, to present more accurate search results. The users then enter the Task Preparation phase once they pick an option.
- **Task Preparation Phase.** Users can enter Task Preparation if and only if they have selected a candidate task. In this phase, users review detailed information of the chosen task and decide if they want to proceed. Users also have access to list management features and the QA module at this phase. If users change their mind, they can go back to Task Search and find another task. Otherwise, they can commit to the task and enter Task Execution.
- **Task Execution Phase.** After entering Task Execution, users are not allowed to change phases, per the competition rule. In this phase, TacoBot walks the user through the instructions step-by-step to help them complete a task with the assistance of the utility module.

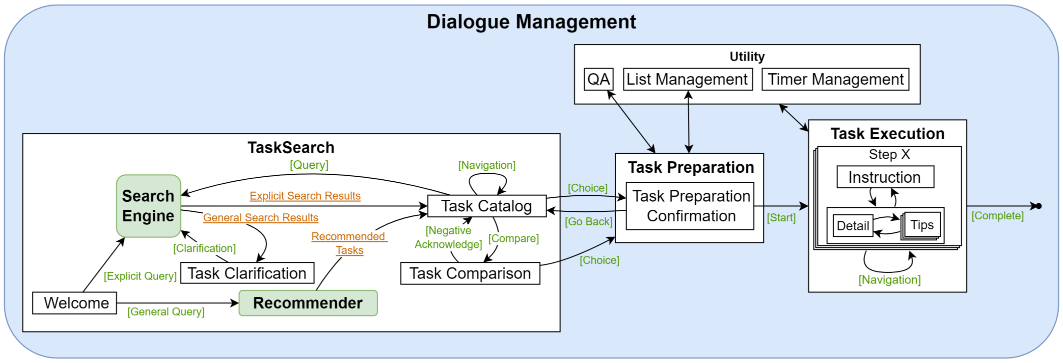


Figure 3: Simplified View of TacoBot's Dialogue Management Module. White boxes represent dialogue states and green boxes represent supporting modules. Bidirectional edges represent reflexive transitions. Green text represents user intent and orange text represents search engine output.

Each step of a task has its own state in this phase. We further break long WikiHow steps into shorter *Instruction*, *Detail*, and *Tips* to make it easier for the user to digest.

- **Utility Module.** TacoBot users can access various utility features, including question answering, help information, as well as Alexa list and timer management through the Utility Module. The utility features are invoked by intents and do not affect the states in the three TacoBot phases.

Upon user input, the dialogue manager performs state transition and choose response generators (Section 6) accordingly. We impose strict conditions for transitions between phases. For example, a specific task must be selected when transitioning from the **Task Search** phase to the **Task Preparation** phase. Within each phase, we assign a dialogue state for each possible conversation turn hierarchically. This hierarchical dialogue state design allows us to define flexible transitions at different levels. For example, users can access the utility module from any state in the **Task Execution** phase. We also maintain a dialogue state history stack to allow users to go back to previous states easily. User intents that do not make valid transitions have no impact on dialogue states. Instead, they trigger contextualized help information that guides users to proceed with their dialogues. As a result, TacoBot provides stable yet flexible dialogue experiences.

5 Search Engine

The task search quality is essential to user experience. We first build a search engine for cooking (from WholeFoodsMarket) and home improvement (from WikiHow) tasks based on elastic search, and further improve the search results with query expansion. A neural re-ranking module is developed to rerank search results for home improvement tasks.

5.1 Query Expansion

One major flaw of using elastic search is that it overemphasizes lexical similarity between the input query and task titles, leading to semantically mismatched search results in many cases. While task titles are usually long and comprehensive, real user requests tend to be short, long-tailed paraphrases of them, making it even more difficult for searching. We develop a query expansion technique to alleviate the impact of such mismatch and improve search result relevance. Query expansion expands task names in user queries by adding related words to improve the recall of search results, including lemmatized verb, nouns, and decomposed compound nouns. For example, the expanded query for *"How to remove spraypaint"* is ["how", "to", "remove", "spraypaint", "spray", "paint"].

5.2 Neural Re-ranking

Since our system shows three search results at a time, hit rate of Top-3 and Top-6 is very important to improve the precision of search results shown to the users. Particularly, for home improvement tasks, the lexical mismatch between user task request and task titles is noticeable. To further improve search performance, we build a neural re-ranking model that brings semantically related search results to the front of the list of candidate tasks. Based on BERT-based model, the re-ranker takes a task request and retrieved task titles as input and assigns a score for each task. We adopt a weakly-supervised list-wise ranking loss (Cao et al., 2007) for training, where one positive and n negative samples ($n = 9$) are used in each step.

To avoid the cost of human annotation, the re-ranker is trained on synthesized task queries via GPT-3 query simulation, as described in Section 3. We propose to collect weak supervision signals from Google's search engine. Specifically, we limit the search space to Wikihow when searching for a home improvement task by appending the suffix, "*site:wikihow.com*", to the task request. The top-3 returned search results are used as positive labels for that task request. We collect hard negative samples by querying elastic search with the same requests and finding any results that do not overlap with Google's top-3 ones.

6 Response Generation

Our response generation module mainly leverages handcrafted conditional rules to fill slots with retrieved data and glue templated segments together. For task-related questions, we build a neural question answering model to detect their relevance and extract their answers from contexts.

6.1 Template-Based Generation

TacoBot stores the response templates as segments of phrases and sentences. The templated segments are carefully curated by native speakers and have several pre-written paraphrases. At composition time, they are selected randomly to generate diverse, human-like responses. We organize the templates and their composition rules according to the high-level states in our hierarchical finite-state machine.

When presenting structured data, TacoBot uses sentence-level templates and substitutes named slots with their data values. For instance, our task preparation response provides descriptive information about the candidate task. To generate this response, the module selects and fills the template based on attributes like ratings, popularity, and estimated time. When presenting unstructured texts, TacoBot uses regular expressions to segment them into short, conversation-friendly pieces and combines them with phrasal or sentence-level templates. During task execution, for example, we index each step with prefixes and append possible commands for users to try in their response.

Alexa List and Timer Service. CoBot also provides interfaces to two kinds of useful Alexa service, list and timer. We recognize that they are essential for users to complete their tasks. Therefore, when users intend to invoke these two kinds of service, we enable TacoBot to generate templated acknowledgements and fill in slots describing its action, similar to how it presents structured data.

6.2 Question Answering

Question answering (QA) is a crucial functionality of task-oriented dialogue systems. In addition to the open-domain EVI system provided by Amazon, TacoBot's QA module has an in-context machine reading comprehension (MRC) module for context-dependent QA, a frequently-asked questions (FAQs) retrieval module for DIY tasks and a rule-based ingredient and substitute QA module for cooking tasks. On top of these QA modules, we build a question type classifier to decide which QA module to call given a question from the user.

6.2.1 Question Type Classifier

We build a question type classifier that classifies a user question into 5 types (`MRC, FAQ, Factual, Ingredient, Substitute`) under a cooking task and 3 types under a DIY task (`MRC, FAQ, Factual`). `Factual` questions are taken by the EVI system and other questions are handled by corresponding QA modules (see below). To better differentiate different types of questions, we concatenate the instruction of the current step (if available) as context with the input question and feed the sequence into a `Roberta-base` classifier. We sample 5,000 questions for each type (see examples in Table 1) in our training set.

Question Type	Example	Context
MRC	Use what tool to blend?	add a 14-ounce can of sweetened condensed milk, unsweetened natural cocoa powder... use a whisk to blend the ingredients until they're completely mixed, and set the bowl aside. it's normal for the mixture to become very thick as you mix it.
Factual	How much is 14 ounce in gram?	
FAQ	How should I know the ingredients are completely mixed?	
Substitute	I don't have condensed milk, can I use something else?	
Ingredient	How much cocoa powder do I need?	

Table 1: Examples of different types of questions.

6.2.2 Context-Dependent QA

We started with UnifiedQA (Khashabi et al., 2020), a pre-trained language model for robust cross-domain QA, and observed two issues in real user conversations: (1) it often fails to detect *unanswerable* questions; and (2) it sometimes generates unrelated hallucinated answers. For example, when the user asks *"Where can I buy peanut seeds and soil?"* during executing "*grow nuts*", the model generates *"peanuts are sold at my local stores"* as the answer. To solve these issues, we annotate an in-context QA dataset and fine-tune an extractive QA model.

Data Annotation. Following the annotation protocal of SQuAD (Rajpurkar et al., 2016, 2018), we first sample 1832 paragraphs[4] as context from WikiHow articles for cooking and home improvement tasks. Then, we ask 15 graduate students to create up to 3 question-answer pairs for each paragraph. The annotated answers are either sentences in the context for answerable questions or a special '[No Answer]' token for unanswerable questions. In total, we obtain 5183 QA pairs including 752 *unanswerable* questions (Table 2).

# of samples	context	answerable QA	unanswerable QA
	1832	4431	752
avg len (# of words)	context	question	answer
	93.9	9.76	17.9

Table 2: Statistics of our annotated WikiHow QA dataset.

Model Fine-tuning. To avoid hallucination, we use Roberta-base (Liu et al., 2019) to develop an extractive QA model in two stages. We pre-train our model on SQuAD2.0 (Rajpurkar et al., 2018) before fine-tuning on our annotated QA dataset. Taking the observation that user may ask questions about previously shown steps, we augment the context by concatenating the current step with the previous n steps ($n = 2$) for training and inference.

6.2.3 Context-Independent QA

FAQ Module. Frequently-asked questions (or Expert QAs) are important knowledge sources in WikiHow articles. Since the questions are raised by the real users and answers are from human experts, such FAQ pairs can reliably answer similar questions from TaskBot users. We collect QA pairs from the *Community Q&A* section of WikiHow articles and use them as our FAQ knowledge source. TacoBot's FAQ QA module is a retrieval module based on the cosine similarity between question embeddings, which are produced by a sentence-BERT (Reimers and Gurevych, 2019)[5] encoder. At inference time, we use a similarity threshold of $t = 0.75$.

Ingredient and Substitute QA Module. To support questions regarding ingredients in cooking tasks, we extract the ingredient mentioned by users through a high-recall string matching mechanism against the list of ingredients in the chosen recipe. When users don't have a particular ingredient, TacoBot could try to offer substitution suggestions if possible. To that end, we collect a substitution data set which covers 200 frequently used ingredients.

7 User Engagement

While a taskbot is primarily a task-oriented system, we observe early in the competition that the bot was frequently used as an interesting novelty to experiment with. In a study of 200 randomly sampled conversations in the quarterfinals, we find that of the 149 conversations in which a task was initiated, approximately 44% of the tasks were sample tasks suggested in the welcome prompt. Further, we find that in about 10% of the conversations, users attempted to initiate non-task related "chatting" with the bot, frequently asking personality-related questions like *"What's your favorite food?"*. These findings lead us to the conclusions that (1) recommending interesting tasks is likely to improve user satisfaction, and (2) more lively responses can make interactions more enjoyable. Thus, we try to improve user engagement by upgrading our example tasks to curated "favorite" tasks and infusing personality traits into TacoBot's responses.

TacoBot's Favorites. We implement a "favorites" feature where we hand-select tasks to recommend to the user. These tasks are selected based on the quality of the content, seasonality, and customer ratings, but also their relevance to a set of predetermined personality traits and hobbies we decided upon for our bot. For example, we decide that our TacoBot's interests include plants, animals,

[4]A paragraph corresponds to a step in a WikiHow article.

[5]`https://huggingface.co/sentence-transformers/all-mpnet-base-v2`

crafts, and home decor and therefore most of the "favorite" tasks are on these topics. These tasks effectively boost the average user rating for conversations in which users accessed the favorites feature in comparison to conversations in which they did not. Additionally, we plan to further introduce hand-crafted featured articles, which are Wikihow articles rewritten by our team members with extra fun facts, tips and more interesting language. We expect these featured articles to provide users with more engaging conversations and pique their interest in topics unique to our bot.

Improving Rapport. In order to make interactions more engaging and increase user trust, we implement three features: redesigning response templates to be more varied and personable, improving the context awareness of responses by making them dependent upon the user's time of day, and creating an attractive visual interface for users with screens.

8 Automated Test Suite

Automating tests for task-oriented dialogue systems is especially difficult because TaskBots (1) deal with diverse inputs and outputs and (2) rely on third-party services. Nonetheless, the complexity of TacoBot raises our awareness of testing. Therefore, we write an end-to-end test suite from scratch. Our end-to-end tests allow us to identify issues in the system efficiently and fix them before deployment, leading to stronger ratings in the semifinals.

Keyword-Based Testing. Typical tests compare an expected result with an actual result and fail if they are different. Because TacoBot has diverse responses for each dialogue state, exact string matching is too rigid for testing. Instead, we tested for the presence of keywords that all appropriate responses must contain. Meanwhile, our tests forbid a different set of keywords that any fallback responses would have. For example, we ensure that the response to "tell me your favorites" includes the words "recipe," "task," and "favorite" and does **not** include "sorry" **nor** "don't understand."

Taco Monitor. We also build an internal website, Taco Monitor, to review all user interactions and identify any issue in near real time. For important problems we want to eliminate in future dialogues, we add the entire conversation into our test suite. For instance, we observed that when users said "cancel", TacoBot repeated its last response, instead of reminding the user of unable to cancel per the competition rule and prompting help information. By adding one of those conversations as a test case, we prevented the bug from happening again.

9 Results and Analysis

9.1 System-level Performance

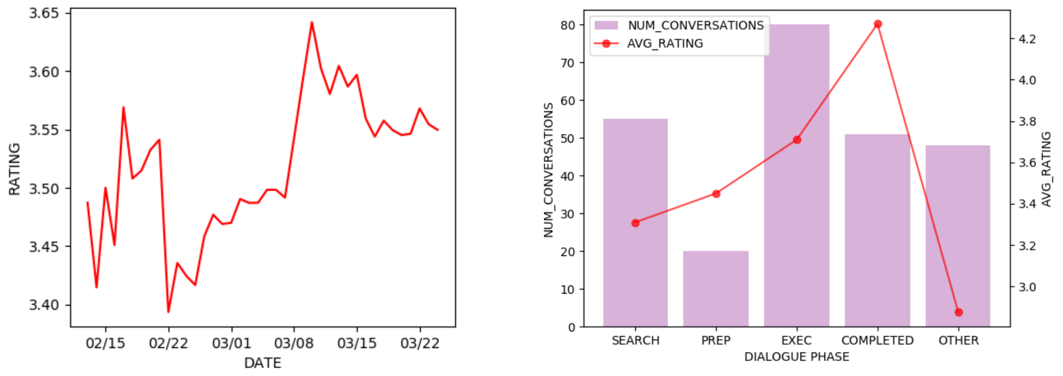


Figure 4: User Ratings from 02/07/22 to 03/24/22. Left: TacoBot's average rating during the semifinals. Right: The number and average rating of conversations by dialogue phases (Section 4) they end in.

Based on our analysis of the sampled quarterfinal conversations (Section 7), we hypothesize that user ratings positively correlate with their progress on the tasks. Thus, we prioritize improving two functionalities of TacoBot, finding relevant search results and offering engaging recommendations, to increase the number of users who start their tasks. As a result, we observe an increase of TacoBot's average rating and a correlation between user ratings and the phases they end in, shown on the left and right of Figure 4, respectively.

Specifically, 55 conversations end in the Task Search phase with an average rating of 3.31, mostly due to irrelevant search results. In contrast, if TacoBot presents satisfying candidate tasks, only 20 conversations (average rating: 3.45) are suspended in the Task Preparation phase, and users of 131 conversations proceed to start their tasks. Among them, users of 80 conversations stop in the middle of a task with an average rating of 3.71, and 6 of them resumed and completed the tasks in a new dialogue session. 45 more conversations record task completion within one session. The average rating of all 51 conversations with tasks completed is 4.27. However, the remaining 48 conversations only have an average rating of 2.88. Most of these conversations include serious exceptions, such as no search results returned, inappropriate task requests, and unexpected user behaviors like chatting.

9.2 Module-level Performance

Natural Language Understanding. We compare our three neural models for NLU (intent recognition, task name extraction, and domain classification) with the built-in interaction model of Alexa Skill Kit (ASK). The evaluation dataset contains 1015 unique utterances collected from real user conversations between 09/16/21 and 12/07/21. We measure the performance on intent recognition and domain classification with accuracy and that on task name extraction with exact match (EM) and Span F1 (Table 3). In summary, our neural models consistently outperform ASK on all three tasks.

	Intent Recognition	Task Name Extraction		Domain Classification
	Accuracy	EM	Span F1	Accuracy
ASK	63.5	30.2	58.1	61.1
Neural Models	**78.7**	**96.3**	**98.6**	**93.3**

Table 3: Results for Our Neural NLU Models.

Search Engine. To measure our search engine's performance, we use real user queries and GPT-3 simulated ones to construct an evaluation dataset. For each query, we retrieve results from Google search and label all relevant candidates as positive. We collected 700 test queries with at least one positive result per query and use hit rate in top-k (HIT-k) to evaluate our search engine.

As shown in Table 4, query expansion improves HIT-3 and HIT-6 by around 7%, compared to the baseline using unmodified queries. We also calculate HIT-25 with query expansion to estimate an upper bound for neural re-ranking. Augmented with the re-ranker, our search engine can closely approximate the upper bound with high HIT-3 (76.9%) and HIT-6 (78.9%). Particularly, the overall HIT-6 is only 2.8% lower than the upper bound (HIT-25), indicating most of the time our re-ranker can get at least one positive in top-6 as long as there is one in the initial top-25 list. Note that we use the DistilBERT as the backbone of the re-ranker to reduce inference-time latency.

Also, we split the test set based on whether the initial search results with only query expansion contain at least one positive in top-k (Easy-k) or not (Hard-k). This split shows that our re-ranker can largely improve the search performance on the hard sets with a limited loss on the easy sets.

Question Answering. We tested the two neural models in our QA module, question type classifier and machine reading comprehension model, with manually annotated test sets. For the question type classier, we evaluated it with 500 questions and observed an overall accuracy of 94%, which indicates the classifier is proficient at deciding the question types.

For the machine reading comprehension (extractive QA) model, we evaluate it on two test sets: (1) our own annotated test set and (2) questions in real TacoBot conversations (12/10/21 - 02/08/22) with answers annotated by a team member. As shown in Table 5, UnifiedQA and Roberta achieve comparable performance on *answerable* questions. However, UnifiedQA performs much worse in identifying unanswerable questions on both test sets. This is reasonable as UnifiedQA was pretrained on data sources which do not contain many unanswerable questions. To provide better

	Overall	Easy-3	Hard-3	Overall	Easy-6	Hard-6
Upper Bound (HIT-25)	81.7	100	51.9	81.7	100	37.6
	HIT-3			HIT-6		
Original Queries	55.7	-	-	63.4	-	-
+ Expansion	62.0	100	0	70.7	100	0
+ Expansion + Re-ranking	**76.9**	96.3	**45.1**	**78.9**	97.8	**33.2**

Table 4: Results of Our Query Expansion and Neural Re-ranking. Easy-k represents the set of samples where at least one positive is found in the initial top-k, while Hard-k is the set with no positive found in the initial top-k. Upper bound (HIT-25) and initial top-k are based on query expansion.

user experience, it is more advisable to give no answer than some random answer. Thus, we adopt Roberta, an extractive model, as the backbone of our QA module.

	Test set (team created)		Test set (from real user)	
	Answerable (436)	Unanswerable (84)	Answerable (5)	Unanswerable (154)
UnifiedQA	39.3	5.9	10.3	22.4
+finetuning	**69.9**	49.2	72.6	43.2
Roberta	40.1	48.8	40.3	88.3
+finetuning	69.7	**69.1**	**72.3**	**88.9**

Table 5: Accuracy (Exact Match) of QA Models. Number in parenthesis means test set size.

10 Discussion and Future Work

Until the semifinals, TacoBot has achieved a 42.0% relative increase in average rating, from $2.50/5.0$ to $3.55/5.0$. In the semifinals, we validated our hypothesis that user satisfaction correlates with accurate search results and attractive recommendations. Nevertheless, TacoBot can be further improved on multiple fronts.

On the one hand, better conversational search and recommendation are required to secure basic user satisfaction. After the semifinals, we introduced a single-turn clarification for recipe searches, but it is still less flexible and limited to diet constraints and cuisines. Instead of increasing the number of clarification questions, it would be interesting yet challenging to enable TacoBot to have a mixed-initiative chitchat around the user's initial query while extracting search constraints and user preferences. On the other hand, we envision TacoBot to be more engaging and accessible. For example, incorporating knowledge-grounded generation models to augment the curated templates would potentially make the responses more conversational and diverse. Also, adding more commands such as volume and pace control could further enhance TacoBot's accessibility.

Finally, the evaluation of TaskBots is a grand open challenge. While Section 9.2 provides module-level performances on our constructed datasets, it is hard to measure the impact of each module on the overall system in real conversations. So far, the performance of the overall system is only evaluated by one single user rating per dialogue, making it harder to measure the correlation between each component and user satisfaction. Moreover, as an evaluation metric, average user rating requires a sufficiently large scale to reflect the TaskBots' overall performance and to conduct A/B tests. When the scale is small, such as less than five per day in the earlier stage of the competition, even one noisy rating could hurt the accuracy of average rating badly. Therefore, we believe it is worthwhile exploring other novel methods (automatic and interactive) to assess the quality of TaskBots.

Acknowledgments

We would like to thank Amazon for financial and technical support as well as colleagues in the OSU NLP group for their valuable comments. We specially thank Yu Gu for his contribution that helped take this off the ground.

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, pages 129–136.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

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

Chandra Khatri, Behnam Hedayatnia, Rahul Goel, Anushree Venkatesh, Raefer Gabriel, and Arindam Mandal. 2018a. Detecting offensive content in open-domain conversations using two stage semi-supervision. *CoRR*, abs/1811.12900.

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. 2018b. Advancing the state of the art in open domain dialog systems through the alexa prize. *CoRR*, abs/1812.10757.

Varun Kumar, Ashutosh Choudhary, and Eunah Cho. 2020. Data augmentation using pre-trained transformer models. In *Proceedings of the 2nd Workshop on Life-long Learning for Spoken Language Systems*, pages 18–26, Suzhou, China. Association for Computational Linguistics.

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

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. In *ACL*, pages 784–789.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. In *EMNLP*, pages 2383–2392.

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. 2019. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. *CoRR*, abs/1909.05855.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Kang Min Yoo, Dongju Park, Jaewook Kang, Sang-Woo Lee, and Woomyoung Park. 2021. GPT3Mix: Leveraging large-scale language models for text augmentation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2225–2239, Punta Cana, Dominican Republic. Association for Computational Linguistics.