Tartan: an LLM Driven SocialBot

Liangze Li, Ziyuan Liu, Li-Wei Chen, Ta-Chung Chi, Alexander I. Rudnicky
School of Computer Science, Carnegie Mellon University
{liangzel, ziyuanl3, liweiche, tachungc, air}@cs.cmu.edu

Abstract

We describe our work to date on using large language models (LLMs) in our Alexa Prize Socialbot. Our work has laid the groundwork for looking more closely at using LLMs in a conversational system. We also analyze common patterns in conversations our bot has had with users.

1 Introduction

SocialBots are required to handle a diverse range of domains, for example, music, sports, or fashion. At the same time, SocialBot interactions need to be managed at other levels. On the content level, the bot must be aware of conversational history, assertions made, and topics covered [1]. On the other hand, the bot also needs to handle other functions such as requests for repetition, which we term meta-conversation. This set of requirements makes the task of a SocialBot more complex than just using a single LLM to handle everything. A critical advantage of LLMs is that they free us from the need to develop and maintain large stores of domain-specific information. However LLMs still have gaps in their domain knowledge; for example, unknown topics need to be detected and deflected using an appropriate strategy.

Our team strongly believes that using LLM for open-domain chat is the right solution. We, therefore, develop a system that eliminates the need for manual intervention in conversation generation. However, this was not enough. We discovered that a certain degree of manual intervention, such as designing instructions, was still necessary. We deemed this type of high-level programming acceptable as it did not involve procedural elements and successfully modified the behavior of a model.

2 System Design and Architecture

We introduce the architecture of Tartan SocialBot in this section. An overview is presented in Figure 1.

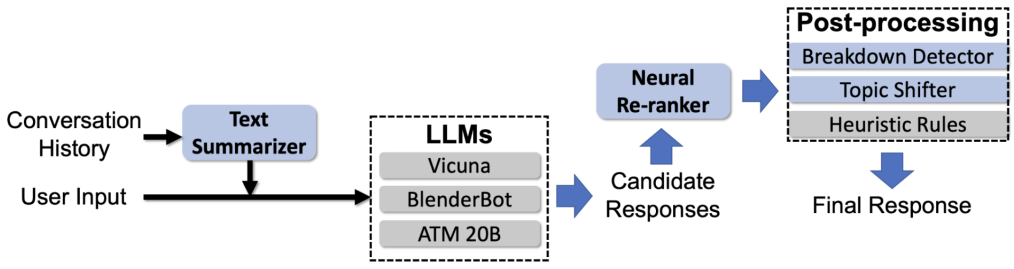

Figure 1: Overview of the Tartan SocialBot.

5th Proceedings of Alexa Prize SocialBot Grand Challenge (Alexa Prize 2023).

2.1 Large Language Models

First, we experimented with different LLMs to figure out the optimal configuration. For each LLM below, we list the issues we found and propose potential solutions.

Alexa Teacher Model (ATM5, ATM20). ATM excels at providing descriptions and information on a given topic, as well as question answering. Nonetheless, it has the tendency to produce short responses with only moderate engagement. For instance, when given "I like to eat pizza." as input, the ATM model responds with "Really? Good to know that ". A better response would be to follow up with questions such as "Do you have a specific preference for pizza?" so that our system can keep the ball rolling. (This is an example of a meta-conversational function.) The style of the generated responses can be improved; In many contexts, the ATM model conveys information in a non-conversational manner. For instance, when a person suggests talking about "forests", ATM will simply return the definition of "forest". In fact, the person was offering a topic shift.

Vicuna. Vicuna is likely the best-performing model out of the three LLMs we deployed; it was specifically fine-tuned on human conversational feedback data. However, there is no free lunch: with superior performance comes high response latency. We attempted to host the 13B variant in the beginning, but the GPU on a g5.8x EC2 instance failed to support the high memory consumption. We tried to resort to some 8-bit quantization techniques in the hope of reducing memory usage. This time we could host the 13B model, but the latency was still far from acceptable. We had no choice but to downgrade the model to the 7B one, which struck a balance between latency and generation quality.

There is still one lingering issue: Vicuna sometimes generates extremely long responses, which hit the 2-second timeout limit we set for our system. While we attempted to prompt it to generate shorter responses, lengthy responses still occur from time to time. When this unfortunate event happens, long responses are automatically dropped. We also had to set a fixed context window size on the conversation history to prevent GPU out-of-memory errors.

BlenderBot. The main issue of BlenderBot is hallucinations and its uncontrollable persona. We observed that BlenderBot often came up with a hallucinated persona, for instance: "I have two sisters and three brothers." Not only is it troublesome due to the contradiction with its existence as a chatbot, but it also induces extra efforts to maintain a consistent persona throughout the conversation. While efforts are made to resolve this issue by prompt engineering (making Blenderbot self-identifies itself as a bot), no significant difference is observed in the persona of BlenderBot. We hypothesize the reasons to be two-fold: First, the dataset BlenderBot was trained on contains mostly human-to-human conversations, apparently leading to a domain mismatch. Second, the original BlenderBot incorporates several additional components including the backstory module responsible for maintaining a consistent persona. We were not able to incorporate this into our system. It's interesting to note that a straight conversational LLM still needs external functions to produce the right behaviors.

2.2 Mixture of LLMs

We use Alexa's BERT re-ranker to enhance the quality of our LLM-generated responses. For each turn, our system outputs the LLM with the highest-ranking response from the BERT re-ranker. Over a 7-day period, we calculated the percentage of instances that a specific LLM was selected. BlenderBot was the most frequent at 54%, followed by ATM at 25%, and Vicuna at 21%. Notably, Vicuna was chosen the least often. This may be attributed to the possibility that Vicuna was trained using data sufficiently different from Alexa's BERT re-ranker, producing a training data mismatch.

3 Reranking Experiments

LLMs that we experimented with exhibited problems such as repetitive cycles and logical contradictions. These problems are inherent especially in BlenderBot. We decided that part of the solution would be to train a conversation reranker that would filter responses that violated good conversational structure. Therefore, we focused on the problem of creating an improved conversation reranking. Our approach uses parameter-efficient knowledge distillation from a larger language model, ChatGPT, without leaking our conversational data.

3.1 The Dataset

Traditionally, training a conversation reranking model is a non-trivial task, due in no small part to the complexity of the structure of workable conversational data [2]. The multiple speakers and complex conversations lead to many possible combinations of dialogue history and current context. This results in a large search space for reranking. Therefore, conversation reranking must involve fine-grained and context-specific criteria; these annotated data. Collecting and annotating such data is resource-intensive, making it difficult to obtain large-scale training datasets.

We used the Daily Dialog (DD) [3] and PersonaChat (PC) [4] corpora to develop a custom dataset. An overview of the extraction process is shown in Figure 2. Since we intend for the resulting dataset to be used to train a ranking module for our BlenderBot-powered SocialBot, our goal was to have a collection of conversations with the last utterance generated by BlenderBot, so the domain will match with SocialBot conversations during inference. We augmented these conversation datasets by making progressive versions of each sample conversation (i.e, turns 1-2, 1-3, 1-4, ...). BlenderBot provided a response at each non-starting timestep. For example, for a conversation of 4 utterances, we construct 3 samples ending up to the 2nd, 3rd, and 4th utterances. BlenderBot then generated a succeeding response. Note that all context utterances are from DD, with no BlenderBot sentences. From the 13,118 conversations in the original DD dataset, a total of 89,861 samples were created.

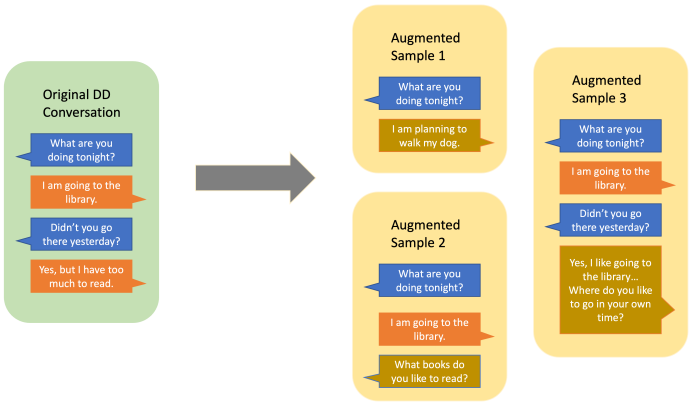

Figure 2: An illustration of the process by which the Daily Dialog dataset is augmented to create the dataset for ranking module training. For a DD dialog with n utterances, we generate $n-1$ distinct samples for training. In the figure, the blue and orange boxes mark original conversation speakers, while the golden box marks BlenderBot filled-in response at each timestep. Notice how the BlenderBot generation at later timesteps relies solely on the original DD content, and does not rely on its generated results for the earlier timesteps. This figure visualizes this process over a made-up toy example for illustration purposes and not an actual conversation in the dataset.

3.2 Knowledge Distillation

Knowledge distillation was achieved by a prompt to ChatGPT for scoring the overall quality of a conversation. Each conversation in our augmented dataset has the last utterance generated by BlenderBot, the rest came from the original Daily Dialog item, considered the ground truth. We conjectured that the ChatGPT scoring would be indicative of BlenderBot response quality (using the DD history).

We convert a conversation to a string by prepending speaker markers (i.e. "A: ", "B: ") and concatenating all utterances. To the end of this string, we append the instruction: "On a scale of 1-4, concisely rate the quality of this conversation. One digit only." This constitutes the query string for

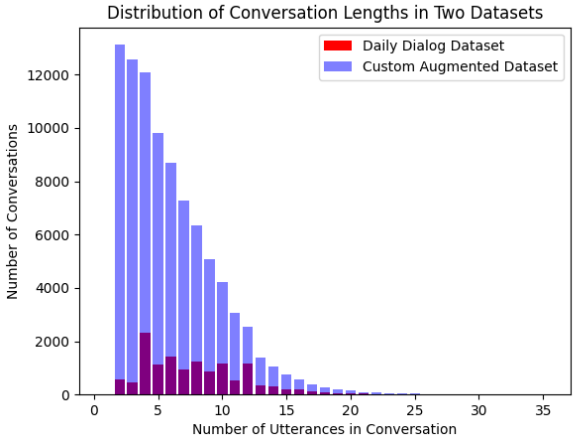

Figure 3: Distribution of conversation lengths in the sourced augmented dataset and the original Daily Dialog dataset. Shorter samples outnumber longer samples in the augmented dataset, because for each DD conversation, we create an augmented sample from every subsequence of that conversation which starts with the first utterance. Data collected over Feb-Aug 2023.

ChatGPT. We found that with this prompt, ChatGPT returns a single digit in the four-point scale [1]. We treated this score as the "gold standard" reference.

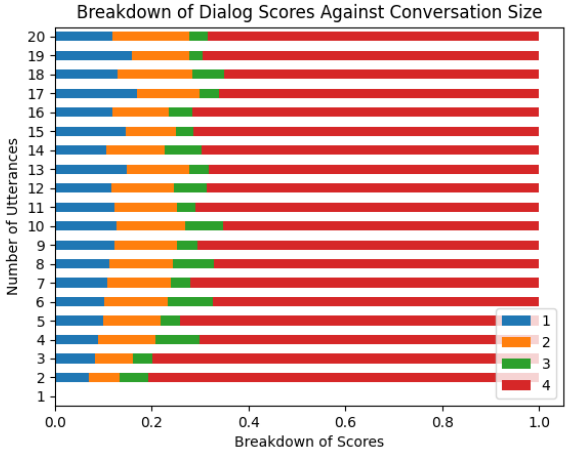

Figure 4: Distribution of the scores given by ChatGPT to conversations. A majority of these are the full score. Shorter conversations tend to receive higher scores, which might be relevant to the logical fallacies and persona issues inherent to BlenderBot. Conversations with more than 20 utterances are not visualized because of the sparsity of such samples. Data collected over Feb-Aug 2023.

We asked human annotators to perform the same task on a subset of the data. Five different participants annotated 120 selected samples on the same scale as ChatGPT. For each sample, we took the average of the human scores for each item and compared it to the ChatGPT score using Spearman correlation, obtaining $\rho = 0.563$ and $p\_value = 2.26 \times 10^{-11}$. This indicates that the ChatGPT scores are significantly correlated to human labeling, validating our initial assumption.

[1]During initial experiments, we tried the 6-point scale. But in that system, ChatGPT hardly produced any 5's, and not many 2's. Therefore, we adopted the 4-point scale by merging the lowest-frequency scores with neighboring values to balance the frequency of different scores.

3.3 Adapter Tuning

Our objective is to train a model capable of giving scores similar to ChatGPT scores. If we do that, we have distilled knowledge from ChatGPT for our domain. We use Adapter Tuning [5] to provide a lightweight alternative to fine-tuning. These are designed as compact feed-forward neural networks that are inserted between the layers of an existing transformer model. Inserts provide for effective transfer learning using fewer parameters. It is also computationally efficient and enables modular usage. In training, the pre-trained model weights are frozen, and only the adapter weights are updated. This makes training a larger model possible with less data than fine-tuning.

For the model backbone, we use RoBERTa [6]. It has the same architecture as BERT[7], but has modifications to the key hyperparameters and to embeddings. This makes it more versatile in downstream tasks. We use the curated conversation scoring dataset for training, with one modification: we balance the number of low scores with the number of high scores. Because ChatGPT gives predominantly high scores, if we used all data directly in training, the resulting model would be biased by the high scores. Accordingly, we randomly sub-sampled full-score instances such that the frequency of low scores (1 and 2) was roughly equal to the frequency of high scores (3 and 4).

We split the dataset into training, validation, and testing sets in the ratio 70:15:15. We use an initial learning rate of 3e-4. Given the amount of training data, the model quickly converges within 5 epochs. The resulting model achieves an accuracy of 59.6% with F1 of 0.591.

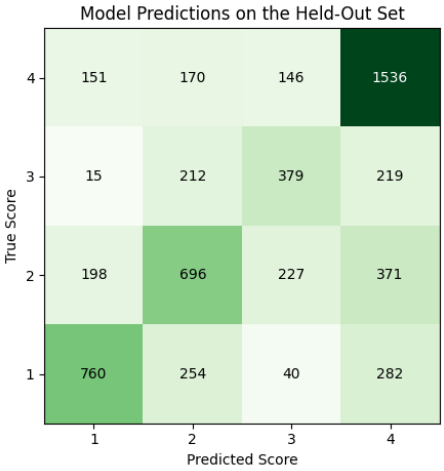

Figure 5: The model's prediction on the held-out set. Extreme scores (1 and 4) are an easier decision for the model, but middle scores (2 and 3) appear a harder choice. Data collected over Feb-Aug 2023.

We observe that misclassifying a `4` as a `1` is a larger error than misclassifying a `4` as a `3`; classification metrics don't usually capture this. Accordingly, we calculate the deviation in scoring as the difference between the predicted score and the true score. This deviation is distributed with mean 0.175 and std 1.14, which suggests that the model tends to overrate (optimistically) but is overall quite informative.

As a single indicator, we developed an aggregate metric: average squared deviation (ASD), This is the result of taking the square of the deviations and taking the mean across all samples. Formally, over the set $\mathcal{I}$ of evaluation samples,

$$\text{ASD}_{\mathcal{I}} = \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} (\text{predicted\_score}(i) - \text{true\_score}(i))^2$$

A larger deviation means more loss. The square allows us to aggregate over all samples. Using this metric, our model generates an ASD of 1.33 over the held-out set.

3.4 Efficiency and Use

We demonstrate a promising approach to knowledge distillation using a large language model on conversation-related domain-specific information. We leverage the parameter-efficient method of

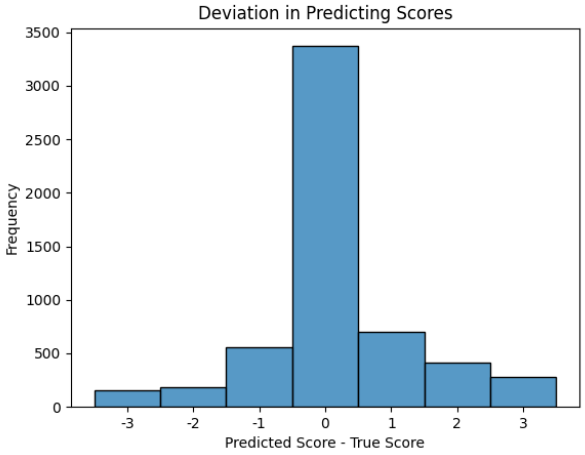

Figure 6: The deviation is defined as the number obtained by subtracting the model-predicted score from the true score. Such deviation is distributed with mean 0.175 and std 1.14. This signals the model has a slight tendency to overrate, but is overall very accurate. The square of the aforementioned deviation measure is distributed with mean 1.33 and std 2.52. Data collected over Feb-Aug 2023.

Adapter Tuning to adjust a pre-trained model to our task (to rate continuations). We show that this combination yields a lightweight pipeline for training domain-specific discriminators for NLP.

We believe that since the model is trained on dialog datasets such as DD, the domain on which the scorer model is trained on should be close to the actual SocialBot conversations. However, putting this trained ranker to use in SocialBot, we find that a large portion of SocialBot conversations are rated a full score. This could be reflecting a domain mismatch between our academic training data and SocialBot conversations. We devise two fixes for this problem: continue incorporating conversational training data across domains, and framing this as a regression task rather than a classification task.

It is also worth noting that we have chosen to use BlenderBot to complete the conversations only in the last turn. In effect, this can be seen as a form of teacher-forcing. However, this assumes the conversation history is always as coherent as the DD corpus. In reality, the user and the SocialBot may deviate off-topic at multiple timesteps in the conversation. This difference is a potential for domain mismatch, which may weaken the model's performance. Therefore, more comprehensive dataset building, which includes completing the dataset at various timesteps, or even with multiple language models, will be advisable.

We expect that this approach can be applied to the other LLMs that we are using to improve overall performance. We also intend to investigate whether the same approach can be used on SocialBot data. The data are noisy but they are from the actual domain. We also intend to determine whether other, smaller, in-house models can achieve performance comparable to ChatGPT on this task, given its limited scope.

4 Common Conversational Patterns

In this section, we explore some common patterns in the conversations that users have had with our socialbot, and consider some potential improvements to the user experience based on these insights.

Given the distinct patterns in utterances at different stages of conversations, we break down conversations into 4 stages for the sake of our analysis, namely:

1. Early Conversation. This stage consists of the user's launch phrase and the socialbot's welcome prompt. Utterances in this stage can often set the tone for the rest of the conversation.
2. Mid-Conversation. Anything between early and late conversation falls into this stage. The most substantive portions of the conversation take place here.

3. Late Conversation. This stage consists of up to the last 4 utterances, excluding any utterances in the early conversation stage, immediately preceding End Conversation stage. Since these utterances lead up to a user decision to end a conversation, they may hint at why a user has decided to end the conversation, such as a breakdown.
4. End Conversation. This stage consists of the last 2 utterances of a conversation.

4.1 Early Conversation

Every conversation begins with the user uttering a launch phrase which connects them to the socialbot, and the socialbot replying with "Hi, this is an Alexa Prize Socialbot" and an initial prompt to kickstart the conversation with the user.

The importance of these first two utterances is highlighted by the fact that around 7.6% of our conversations end immediately afterward, in a phenomenon we call "early abandonment". Naturally, this is not conducive to our user engagement; in an ideal world, our socialbot should be able to engage every user in a substantive conversation. We therefore look into these first utterances in an attempt to elucidate the causes of early abandonment with a view to reducing the early abandon rate.

4.1.1 Launch Phrases

The rate at which conversations are abandoned early differ wildly depending on the launch phrase used, as we can see from Table 1. We can see that "Open Alexa Prize", the official launch phrase used by the Alexa Prize SocialBot Grand Challenge 5, has an early abandon rate of 3.3%, while conversations initiated by a request to talk to a specific chatbot have an early abandon rate of 12.2%.

In our view, the intent of a user connected to the socialbot is a very important factor in determining the likelihood of early abandonment. Intuitively, a user who proactively seeks to talk to a socialbot is less likely to abandon a conversation early, while a user who is not expecting to connect to a socialbot may be the most likely to do so. For instance, a user seeking to talk to a specific chatbot might be disappointed with being connected to an "Alexa Prize Socialbot" (12.2% early abandon rate), while a user requesting a "social bot" will have gotten precisely what they wanted (0% early abandon rate).

Table 1: Early Abandon Rate by common Launch Phrase. Data collected over Feb-Aug 2023. Actual user phrases are hided to protect privacy.

Phrase	Early Abandon Rate	Percentage (%)
Open Alexa Prize	0.033	47.3
phrase 1 (Topic based)	0.171	6.60
phrase 2	0.149	12.2
phrase 3 (Specific Chatbot)	0.122	4.86
phrase 4	0.061	10.1
phrase 5	0.055	17.6
phrase 6	0.041	0.686
phrase 7	0.000	0.677
Average	**0.076**	

Interestingly, we see that when the launch phrase requests for a discussion on a specific topic, the abandon rate is very high, at 17.1%, or nearly 1 in every 5 conversations. In this scenario, the user may have already made up their mind about the topic they want to have a conversation on, and a welcome prompt unrelated to the topic is highly likely to result in early abandonment.

4.1.2 Welcome Prompts

Since the welcome prompt is the first utterance from our socialbot to the user, it is the first opportunity to shape the conversation and impress the user (and the only opportunity to change the course of a conversation at risk of early abandonment). It is possible that at least some conversations subject to early abandonment might have continued had the welcome prompt met the users' expectations or otherwise caught the attention of the user.

We find that in general, using a prompt that suggests a topic for discussion helps better shape the conversation and results in overall longer conversations than using a generic greeting such as "How are you doing today?"

4.1.3 Discussion

Since the launch phrase used by a user may be indicative of their intent at the point of launching the Alexa Prize Skill, it would be wise to develop different strategies to appeal to different user types in our welcome prompts.

Users who launch the bot unintentionally need to be coaxed to remain, if possible - we will investigate strategies for doing so in future work.

Users who request a specific conversation topic should be given a welcome prompt that specifically addresses the topic requested, in a way that would indicate an ability to have an interesting conversation on the topic with the user. We are developing methods to identify these users and their topics of interest, as well as strategies to convey the right amount of enthusiasm and knowledge on the topic to keep the user interested in the conversation.

Users who explicitly request a socialbot conversation are least likely to abandon early, regardless of our welcome prompt. However, the welcome prompt is still relevant in determining the trajectory of the ensuing conversation. It may be helpful if users are steered towards conversation topics that are more likely to result in a long and substantive conversation.

4.2 Mid-Conversation

The Mid-Conversation stage consists of all utterances between the Early and Late conversation stages. Therefore, in a conversation of sufficient length, the most substantive portions of the conversation can be expected to be located here. It is desirable for this stage of the conversation to continue for as many turns as possible - in order for this to happen, we must be able to sustain user interest and avoid breakdowns that might end an otherwise thriving conversation.

Breakdowns Given the limited extent to which we can control the output of the LLMs and the natural flow of conversation from one topic to another, breakdowns occur in conversations from time to time. We will discuss some common breakdowns and the success of our strategies to combat breakdowns in the next subsection.

4.3 Late Conversation

The Late Conversation stage consists of the third- and second-to-last user-bot utterance pairs in the conversation. These represent the run-up to the end of a conversation and may provide insights that can help us develop strategies to prolong conversations and improve user engagement.

User utterances in this phrase are indicative that the termination of the conversation is imminent. Their satisfaction or interest level may have dipped, and they are increasingly inclined to stop talking to the bot. Strategies must be in place to placate users and persuade them to continue in the conversation (without explicitly doing so, of course).

Bot utterances in this stage mostly indicate a breakdown in conversation, or an unsuccessful attempt to manage a breakdown. We will have to consider strategies, such as prompt engineering, to minimize the likelihood of LLMs generating utterances that cause breakdowns, and improve our ability to recover from such breakdowns.

4.3.1 User Utterances

User utterances in the late conversation are markedly shorter than is typical; while the average number of words across all user utterances (apart from launch phrases) is 4.75, the average number of words in user utterances in the late conversation onward is 3.03.

Many of these user utterances in the late stage express frustration, apathy, or a desire to change the topic; these lead to the termination of the conversation once users' interest in continuing the conversation is depleted.

Where user utterances request a change in topic in this stage, many conversations end because the bot is unable to fulfill that request satisfactorily. We need to consider ways to prompt the LLM to be more proactive in suggesting conversational topics that are likely to satisfy the user and move the conversation along, with the knowledge of the contents of the conversation thus far.

4.3.2 Bot Utterances

Table 2: Common Late Conversation Bot Utterances. Data collected over Feb-Aug 2023.

Phrase	Percentage (%)
I don't feel comfortable talking about that. Let's talk about something else.	42.7
Let's talk about something else, would you like to talk about [topic]?	28.8
Let's chat about something else, what would you like to talk about?	6.50
Greetings. What's on your mind?	5.60
Hi, open Alexa Prize!	5.34
okay/ok	3.57
Sure, what would you like to talk about	2.82
Ok, great! What's your name?	1.71
You're welcome	1.62
Is there anything else I can help you with?	1.35

Table 2 summarizes the common late conversation bot utterances. Many bot utterances in this stage either directly lead to conversational breakdowns ("Breakdown Causation"), or represent attempts to manage breakdowns ("Breakdown Management"). Many utterances are also overly short and devoid of substance or initiatve (e.g. "you're welcome", "thank you", "okay" etc.) - they fail to move the conversation forward and leave the onus of continuing the conversation on the user alone.

Breakdown Causation Some bot utterances in this stage are agents of chaos - it is immediately obvious that they directly contribute to a conversational breakdown. While some of these breakdowns are apparent only with the benefit of context (see 4.3.3), others may be much more glaring. A common breakdown is a bot utterance greeting the user as though the conversation just started (e.g. "Hi, open Alexa Prize!", "Greetings! What's on your mind?", etc.) when the conversation has already proceeded for a considerable number of turns.

Breakdown Management As with user utterances, many bot utterances in this stage also suggest a change in the topic (e.g. "let's talk about something else"). Since changing the topic is one of our main strategies for responding to a breakdown in the conversation, the large number of such utterances in this stage may indicate that this strategy is not entirely successful and we may have to reconsider strategies for handling breakdowns.

4.3.3 Contradiction

As discussed earlier in section 2.1, some LLMs are prone to hallucination and have a tendency to generate utterances that make claims in contradiction with information conveyed to users in previous utterances. A key prong of our investigation into conversation management is in effectively identifying such contradictions before contradictory utterances are sent to the user; one strategy could be to use prompting to get another LLM to make the judgment on whether an utterance is contradictory.

4.3.4 Proactive Termination

While it may seem counterintuitive, having the bot proactively end the conversation at this stage may improve the user experience. Some prompts to the bot have been observed to attempt to terminate the conversation, without using a phrase that will immediately end the conversation. Their intent to leave is nonetheless very clear. At the extreme, some users have spent more than 20 turns trying to end the conversation and have ended up extremely frustrated. Other users attempt to trigger functionality from the regular Alexa environment (e.g. asking for time, playing music etc.). In these cases, it may be best for the user experience if the bot quits promptly to avoid further frustration.

4.4 End Conversation

The End Conversation stage consists of the last 2 utterances of a conversation. The last input to the bot is often one of a small set of instructions that immediately exit the bot (phrases such as: "quit", "turn off", "stop", "okay bye", "shut up" etc.). While this may not be very illuminating, it does shed some light on the emotional exchange of the conversation upon exit. For instance, a conversation that terminates with something like "shut up" is less likely to be satisfying compared to a conversation that ends with "Okay, bye". The last bot utterance, which always occurs before the last user utterance, could be useful in determining why the user chose to end the conversation at that particular point in time and can be processed in a similar way as late conversation bot utterances for the purposes of error analysis.

5 Future Work

Fine-tuning with in-domain Data. Our initial conjecture was that LLMs would be able to generate good-quality content for open-domain conversation. We believe this is the case, but the implementation needs to be more sophisticated than what we started with. We would like to investigate in several directions:

LLM ensembles Our current implementation uses 4 LLMs. But the approach allows for larger ensembles. This creates several opportunities. For example, in the current ensemble the imbalance between models should be better understood: is this a problem with the re-ranker, in that it's more disposed to certain kinds of responses, which are not, in fact, the optimal ones? This might be a question of how the re-ranker was trained, such as on a corpus not well aligned with Alexa conversations. Or it can be that the models being rejected in fact are not that good. If so it means that the system could "audition" models to find a (compact) set that gives better conversations. A second direction would be to start fine-tuning on our Alexa data, to try to capture the right characteristics.

We have pointed out that the LLMs used have not been specifically optimized for the type of conversations we are attempting to simulate. While many models claim to have been trained on "conversation" data, in reality, these datasets are often derived from text exchanges such as those found on Reddit or solicited from users providing instruction rather than spontaneous chat. Additionally, datasets that use spoken language interaction are limited, and often only derived from goal-directed dialogues. Even if the models are trained on spontaneous conversations, achieving optimal performance is challenging.

LLMs for different functions It might be possible to use LLMs to provide information for use at the meta-conversation level. We have looked at the problem of contradiction. Users are quick to notice this problem, but LLMs (at least the ones we have available) do not do a good job of adhering to their history. Similarly, a Model can get itself into a loop with the user where the same bit of conversation is recycled. In the past, we would have tried to solve this problem in an ad hoc fashion (say with REs). A better solution might be to use LLM(s) that treat the problem on a more abstract level, like detecting more subtle contradictions (e.g.the exchange about siblings). Models individually tuned to these phenomena (leading to particular repairs) might be the way to do it and open up more possibilities for managing conversations.

References

[1] M. Johnston, C. Flagg, A. Gottardi, S. Sahai, Y. Lu, S. Sagi, L. Dai, P. Goyal, B. Hedayatnia, L. Hu, D. Jin, P. Lange, S. Liu, S. Liu, D. Pressel, H. Shi, Z. Yang, C. Zhang, D. Zhang, L. Ball, K. Bland, S. Hu, O. Ipek, J. Jeun, H. Rocker, L. Vaz, A. Iyengar, Y. Liu, A. Mandal, D. Hakkani-Tür, and R. Ghanadan, "Advancing open domain dialog: The fifth alexa prize socialbot grand challenge," in *Alexa Prize SocialBot Grand Challenge 5 Proceedings*, 2023.

[2] B. Liu, J. Kang, J. Zhao, J. Huang, X. Zhu, Y. Wu, and D. Xie, "On the evaluation of dialogue systems with next utterance classification tasks," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2071–2081, 2016.

[3] Y. Li, H. Su, X. Shen, W. Li, Z. Cao, S. Niu, and S. Zhao, "DailyDialog: A manually labelled multi-turn dialogue dataset," in *Proceedings of the Eighth International Joint Conference on*

Natural Language Processing (Volume 1: Long Papers), pp. 986–995, Asian Federation of Natural Language Processing, 2017.

[4] S. Zhang, E. Dinan, J. Urbanek, A. Szlam, D. Kiela, and J. Weston, "Personalizing dialogue agents: I have a dog, do you have pets too?," 2018.

[5] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly, "Parameter-efficient transfer learning for nlp," in *Proceedings of the 36th International Conference on Machine Learning*, vol. 97, pp. 2790–2799, 2019.

[6] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.