Neural Generation Meets Real People: Towards Emotionally Engaging Mixed-Initiative Conversations

Ashwin Paranjape,* Abigail See,* Kathleen Kenealy, Haojun Li, Amelia Hardy, Peng Qi, Kaushik Ram Sadagopan, Nguyet Minh Phu, Dilara Soylu, Christopher D. Manning

Stanford NLP

{ashwinpp,abisee,kkenealy,haojun,ahardy,pengqi,
kaushik7,minhphu,soylu,manning}@stanford.edu

Abstract

We present **Chirpy Cardinal**, an open-domain dialogue agent, as a research platform for the 2019 Alexa Prize competition. Building an open-domain socialbot that talks to real people is challenging – such a system must meet multiple user expectations such as broad world knowledge, conversational style, and emotional connection. Our socialbot engages users on their terms – prioritizing their interests, feelings and autonomy. As a result, our socialbot provides a responsive, personalized user experience, capable of talking knowledgeably about a wide variety of topics, as well as chatting empathetically about ordinary life. Neural generation plays a key role in achieving these goals, providing the backbone for our conversational and emotional tone. At the end of the competition, Chirpy Cardinal progressed to the finals with an average rating of 3.6/5.0, a median conversation duration of 2 minutes 16 seconds, and a 90$^{\text{th}}$ percentile duration of over 12 minutes.

1 Introduction

This paper describes our socialbot for open-domain conversation, **Chirpy Cardinal**, built as a research platform during the 2019 Alexa Prize competition. During the competition, US-based Amazon Alexa users could give an invocation phrase (such as *let's chat*) to be connected to one of the competing socialbots (chosen randomly). After receiving a minimal orientation phrase at the beginning of the conversation, the user talks to the socialbot (in English) until they decide to end the conversation – at which point, they are invited to provide a rating and comment.

To provide a convincing user experience, an open-domain conversational agent must excel at language understanding, language generation, emotional engagement, memory, world knowledge and conversational planning, among other desirable characteristics – an ambitious goal! Prior work within and outside the Alexa Prize competition has taken the successful strategy of pushing progress along individual skills, and forming an ensemble of sub-systems, each excelling at a singular characteristic while ignoring others. For instance, supporting user initiative in open-domain conversations is extremely challenging, as it requires understanding the countless ways a user can take initiative, and the ability to respond to each of them with specificity. Faced with this difficulty, when it comes to in-depth conversations, many previous dialogue systems rely primarily on bot-initiative, driving users along carefully scripted paths. On the other hand, systems attempting higher user-initiative via non-scripted paths are likely to lead towards shallower conversations. Thus there is a lot of room for innovation and research in trying to simultaneously achieve two or more complementary characteristics; this is a recurring theme throughout this work. Our goal in building this socialbot was

*equal contribution

3rd Proceedings of Alexa Prize (Alexa Prize 2019).

to offer a natural-sounding and emotionally engaging dialogue agent that can talk knowledgeably about a wide variety of topics, while also letting the user take as much initiative as possible.

Initiative – the ability to drive the direction of the conversation – has been studied extensively in the context of task-oriented dialogue. **Mixed initiative** (Horvitz, 1999), in which the user and the bot share initiative, is an important quality of a successful dialogue system, as it provides the user a sense of agency without making them entirely responsible for suggesting new topics and directions. In order to improve on mixed initiative while still providing an acceptable conversational depth, we designed our initial system to rely heavily on system initiative, but at the same time explored several avenues to increase user initiative in a controlled fashion. To support mixed initiative, our system has a global navigational intent classifier (Section 3.1) and entity tracker (Section 3.2), allowing it to track high level topic changes from both the user and the bot. Further, our response priority system (Section 3.3) allows individual Response Generators (RGs) to interject when the user initiates a change of topic.

High-coverage world knowledge is an important component of open-domain conversation – our bot must be able to talk about the diverse range of entities and topics that interest users, particularly if we wish to respect user initiative. We use the Alexa Knowledge Graph, The Washington Post, Reddit and Twitter as sources of up-to-date knowledge in particular domains, while ensuring high coverage by using Wikipedia and Wikidata entities as the foundation of our entity-based conversations (Sections 4.4, 3.2 and 6.3). However, world knowledge must be delivered in a **conversational style** – this is a characteristic that distinguishes a socialbot from a virtual assistant. To achieve this, we finetuned a neural generative model on the TopicalChat dataset (Gopalakrishnan et al., 2019) to obtain a conversational paraphrasing model that adapts external text into a conversational style (Section 5.3).

A socialbot cannot focus solely on external entities – to be truly *social*, it must be able to discuss **personal experiences and emotions**. While ELIZA-like systems (Weizenbaum et al., 1966) attempt this via templated repetition of user phrases, they lack the naturalness and depth of real human conversations. Our Neural Chat module (Section 5.2) invites the user to share their everyday experiences and current emotions, and uses a neural generative model to respond empathetically. With it, we attempt to have a deep, sustained and emotionally engaging conversation about a user's lives. In addition, our Opinion module (Section 5.4) allows the user to express their feelings by expressing their likes and dislikes. To foster a reciprocal atmosphere, our bot also shares its own distinct feelings, experiences and opinions.

Lastly, we note that the advent of large-scale pretrained **neural generative models** has substantially impacted what is possible in open-domain socialbots. While in the last Alexa Prize competition, none of the top three socialbots used neural generation (Chen et al., 2018; Pichi et al., 2018; Curry et al., 2018), we found current GPT-2 models (Radford et al., 2019) to be a key tool to support our design goals. Neural generation enables natural phrasing and emotional engagement, as well as more flexible responsiveness (e.g., when used as a fallback in Section 5.7), supporting higher user initiative. A limitation of neural generation methods for dialogue is deterioration in quality and consistency over a long conversation, which can be potentially overcome with symbolic constraints. We explore ways to bring the best of both worlds – long term consistency and short term fluidity – together.

Despite being a first-time entrant, at the end of the competition our system achieved a rating of 3.6/5.0, which is within 0.1 of the highest-ranked systems, and is capable of detailed, sustained conversations with interested users (with a 90^{th} percentile conversation duration of 12 minutes 55 seconds). Qualitatively, during in-person interactions with users, we observed that many innovations such as in-depth discussions of everyday life, conversational styling of informational content, and opinionated exchanges were received with expressions of pleasant surprise – indicating our steps were in the right direction. In Section 6, we re-examine the goals we set out to achieve, and empirically analyze our bot's successes and failures. In Section 7, we talk about the challenges we faced, the trade-offs we made, our conclusions and avenues for future work.

2 System Overview

Our overall system design is shown in Figure 1. Our system is built on top of the CoBot framework (Khatri et al., 2018). On each turn, the user's spoken utterance is transcribed by Alexa's Automatic Speech Recognition (ASR) service. The transcribed utterance (which is lowercase, no punctuation) is sent to our AWS Lambda function, which handles the core logic of our bot. AWS Lambda is a

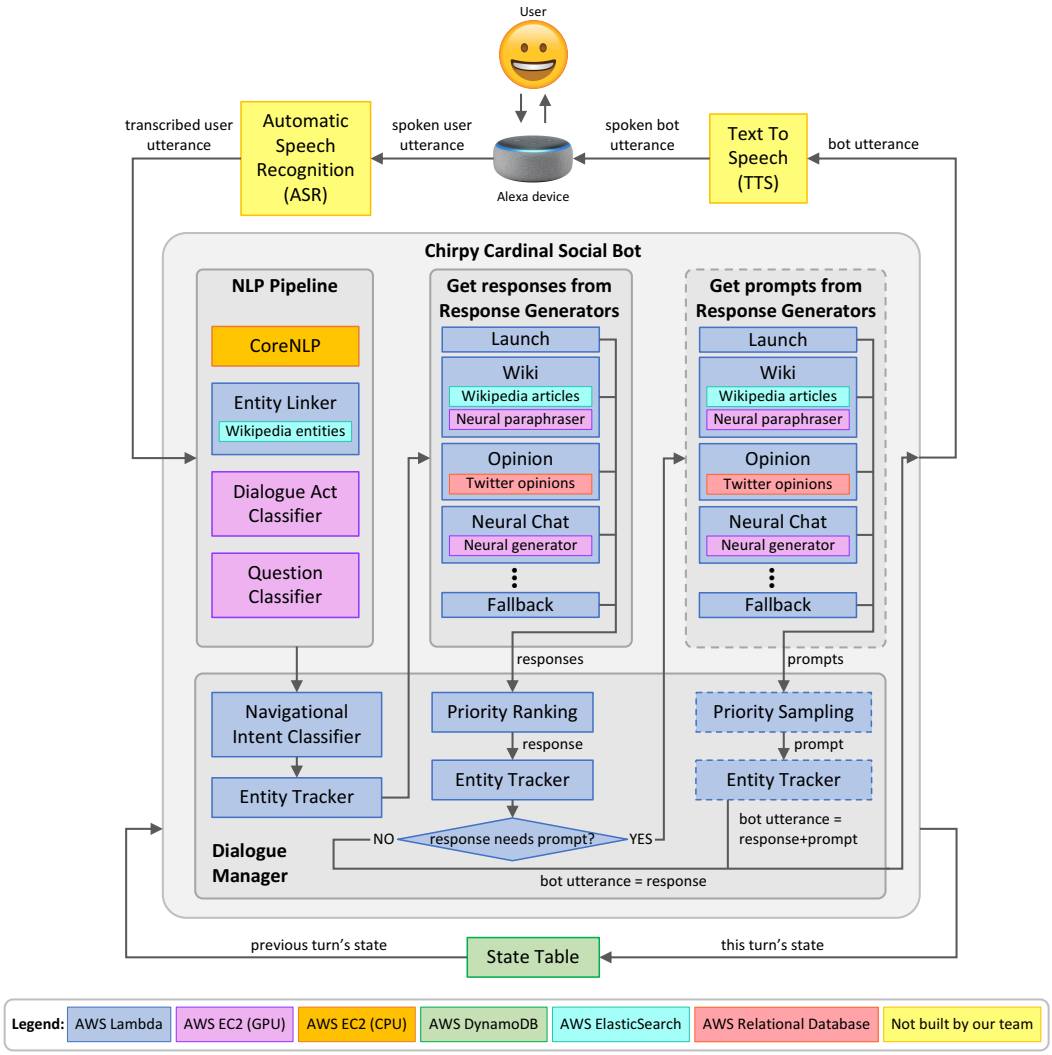


Figure 1: Overall system design.

serverless computing platform, which means that our function is stateless. To preserve information between turns, we store our bot's overall state in an external State Table (see Figure 1), hosted on AWS DynamoDB. At the start of the turn, the previous turn's state is fetched from the table.

We then run the **NLP Pipeline** (see Section 4) – a collection of modules that produce annotations based on the user's utterance and the current state. Modules requiring greater computational resources are hosted on remote EC2 instances, while less-demanding modules are hosted within the Lambda function. The NLP Pipeline is organized as a directed acyclic graph (DAG), allowing modules to use other modules' annotations as inputs. To minimize latency, modules are run in parallel where possible, with each module starting as soon as its inputs are ready.

Next, we analyze the user's utterance to determine whether the user wants to talk about any particular entity (see **Navigational Intent**, Section 3.1), and update the current entity under discussion if appropriate (see **Entity Tracker**, Section 3.2).

We then run our collection of **Response Generators** (RGs), modules designed to handle particular conversational duties, in parallel (see Section 5). Each RG either produces a **response**, or no response (None). If an RG produces a response, it also supplies a **response priority** (see Section 3.3), indicates whether the response needs a **prompt** added from another response generator (see Section 3.4), and specifies what the current entity under discussion should be, if the response is chosen. The **Priority Ranking** module chooses the response with the highest priority, and the Entity Tracker updates the

current entity under discussion accordingly. If the chosen response *does not* need a prompt, it forms the entire bot utterance.

If the chosen response *does* need a prompt, we run our collection of RGs a second time. Each RG either produces a prompt or no prompt (None). If an RG produces a prompt, it also supplies a **prompt priority** (see Section 3.5) and a current entity, as before. The **Priority Sampling** module chooses the prompt by sampling from the supplied prompts, with the probability distribution depending on both the priorities of the prompts and the RGs that produced them. The Entity Tracker updates the current entity again, and the bot's utterance is then formed by appending the prompt to the response.

At the end of the turn, the bot's overall state contains the user's utterance, the conversational history, the NLP Pipeline annotations for the user's utterance, and a state for each individual Response Generator.[2] We write the new state to the State Table, and send the bot utterance to Alexa's Text To Speech (TTS) service, which delivers the spoken bot utterance to the user.

3 Dialogue Management

Our Dialogue Manager handles the high-level logic of tracking which topics we are discussing with the user, and which responses (and prompts) should be used to form the bot's utterances.

3.1 Navigational Intent Classifier

A user has *navigational intent* when they are indicating that they do (*positive*) or do not (*negative*) want to talk about a particular topic. Users might give navigational intent while specifying the topic (*can we talk about minecraft*, *stop talking about minecraft*), or referring to the current topic (*let's discuss this more*, *could you change the subject*), or referring to no topic (*alexa can we talk*, *i don't want to chat any more*). Users sometimes give positive and negative navigational intent in the same utterance (*i don't want to talk about movies any more let's chat about you*). To recognize navigational intent, we use manually-constructed regexes, as they are quite high precision.

3.2 Entity Tracker

For our response generators to work together to discuss different topics smoothly, we must track which entities we are currently discussing, which we have finished discussing, and possible entities to discuss in the future. This is the role of the *entity tracker*. We assume that at any point in the conversation, there is one *current entity*, which is either a Wikipedia entity (see Section 4.4) or None (if we're discussing something that does not have a Wikipedia article, such as Table 1 Turn 3).[3] The current entity is updated at most three times per turn (see Figure 1):

1. **After analyzing the user's utterance**. The entity tracker uses the entity linker's output, which is a priority-ordered list of possible entities mentioned by the user on this turn, along with their scores (see Section 4.4 for details.) If the user expressed negative navigational intent towards the current entity, it is rejected. If the user expressed positive navigational intent towards some topic, we search inside the topic slot in their utterance; the highest-priority entity with score over a low threshold (1,000) is chosen as current entity. If there is a particular type of entity we expect the user to mention on this turn (e.g. if the bot asked *What's your favorite movie?*) and there is an entity with the expected Wikidata category (e.g. *film*) with score over a low threshold (1,000), it is chosen as current entity. Otherwise, if the entity linker has made a prediction with sufficiently high score (over 10,000), it becomes the current entity. If none of these conditions are met, the current entity stays the same.
2. **After choosing the response**. When the RGs provide responses, each RG also specifies what the new current entity should be, if its response is selected by the priority ranker. We update the current entity to be whatever was provided by the selected RG.
3. **After choosing the prompt**. If we get a prompt, we update the current entity similarly.

[2]Each individual RG state contains information required by that RG – for example, it might contain the current node (aka treelet, see Section 5.1) in the RG's dialogue graph, or a list of the utterances and/or entities that have been discussed, to avoid repetition. Each RG maintains its own individual state.

[3]In the future, we plan to generalize the entity tracker to track non-Wikipedia discussion topics.

#	Utterance	Entity	Commentary
1	**User:** let's chat	None	
	Bot: Hi, this is an Alexa Prize Socialbot. I'd love to get to know you a bit better before we chat! Is it all right if I ask for your name? [Launch]	None	Launch gives *FORCE_START* response to begin the conversation.
2	**User:** my name is chris	None	
	Bot: Well it's nice to meet you, Chris! I'm excited to chat with you today. [Launch] I hope your afternoon is going well. What are your plans for the rest of today? [Neural Chat]	None	Launch gives a *STRONG_CONTINUE* response that needs a prompt, which is supplied by Neural Chat with *FORCE_START* priority (scripted transition).
3	**User:** hang out with my friends	None	
	Bot: That sounds great, I love spending time with friends. What will you do together? [Neural Chat]	None	Neural Chat generates a GPT-2 response with *STRONG_CONTINUE* priority.
4	**User:** maybe watch a movie	None	
	Bot: Me too! I love watching movies; I get to learn so much about what the world is like outside of the cloud! Have you seen any movies recently? [Movies]	Film	Movies detects the 'movie' keyword in the user's utterance and gives a *FORCE_START* response (overriding Neural Chat), setting current entity to *Film*.
5	**User:** i saw the matrix	The Matrix	The Entity Linker and Entity Tracker identify *The Matrix* as the current entity.
	Bot: Nice! Did you like The Matrix? [Movies]	The Matrix	
6	**User:** i loved it neo is amazing	The Matrix	
	Bot: Oooh, yeah, I agree. Hey, isn't Keanu Reeves in that movie? What do you think about Keanu Reeves? [Movies]	Keanu Reeves	Movies sets current entity to *Keanu Reeves*.
7	**User:** i want to talk about something else	None	Navigational Intent detects *negative intent*, so Entity Tracker rejects the current entity.
	Bot: OK, no problem. [Movies] There's actually something else I wanted to ask you about. What's your favorite animal? [Categories]	Animal	Movies acknowledges and asks for a prompt. Categories gives a *GENERIC* prompt.
8	**User:** i love cats	Cat	Entity Linker and Entity Tracker identify *Cat* as current entity.
	Bot: Good to hear you like cats. I have to be honest though, I'm not a big fan of cats. I feel like cats don't respect my personal space, but I would love to hear why you like cats? [Opinion]	Cat	Opinion gives a *CAN_START* response about cats.
9	**User:** hmm i love cats because they are fluffy	Cat	
	Bot: That make sense. Now that I think about it, one good reason to like cats is that they purr and I definitely need that kind of positive feedback. [Opinion] Wanna know something interesting about cat? [Wiki]	Cat	Opinion finishes talking. Wiki gives a *CURRENT_TOPIC* prompt about *Cat*.
10	**User:** you are not very smart	Cat	
	Bot: I know you feel frustrated. I'm always trying to get better. [Offensive User] I've been listening to some new music today and I wanted to chat about instruments. If you were a musical instrument which one would you be? [Music]	Musical instrument	Offensive User gives a *FORCE_START* response to the critical utterance, then [Music] provides a *GENERIC* prompt to steer the conversation in a new direction.
11	**User:** what do you find interesting	None	The user gives an unexpected response.
	Bot: I like the violin, but I'm more of a classical music player. [Neural Fallback] I remember you mentioned Neo. Would you like to talk more about it? [Wiki]	Neo (The Matrix)	Neural Fallback gives a GPT-2-generated response, which is selected as no other RG (except Fallback) has a response. Wiki gives a *CONTEXTUAL* prompt.
12	**User:** sure	Neo (The Matrix)	
	Bot: Neo has got into so many adventures! What are some adventures that you've enjoyed? [Wiki]	Neo (The Matrix)	Wiki asks an open-ended question (specific to movie characters) about *Neo (The Matrix)*.
13	**User:** morpheus teaching jujitsu to neo	Neo (The Matrix)	
	Bot: I liked that Neo and Trinity were able to rescue Morpheus from a building protected by armed guards and agents. Morpheus has done so many amazing things! What are some of your favorite Morpheus moments? [Wiki]	Morpheus (The Matrix)	Wiki gives a response (generated by the GPT-2 conversational paraphrasing model), sets the current entity to *Morpheus (The Matrix)*, and asks another open-ended question.
14	**User:** i want to stop talking	None	We detect *stop intent*; the conversation ends.

Table 1: An example dialogue. To respect users' privacy, this is not a real user conversation.

Response Priority	Meaning
FORCE_START	This inactive RG should take control (e.g., Table 1, Turn 4), or override, such as handling offensive user utterances (e.g., Table 1, Turn 10).
STRONG_CONTINUE	This active RG can continue the conversation with a good next response (e.g., Table 1, Turn 2). Only a *FORCE_START* can override it.
CAN_START	This inactive RG can potentially take control (e.g., Table 1, Turn 8), but should not interrupt a *STRONG_CONTINUE*.
WEAK_CONTINUE	This active RG can continue the conversation but its next response is of poorer quality. It should be overridden by any available *CAN_START*s (or higher).
UNIVERSAL_FALLBACK	Only used by Fallback and Neural Fallback RGs (e.g., Section 5 and Table 1, Turn 11)

Table 2: Response Priorities (ordered by descending importance)

Prompt Priority	Meaning
FORCE_START	This RG should take control. This is mainly used for scripted transitions (e.g., Table 1, Turn 2).
CURRENT_TOPIC	This RG has a prompt that talks about the current entity (see Section 3.2 and Table 1, Turn 9).
CONTEXTUAL	This RG has a prompt that does not talk about the current entity, but that is conditioned on the conversation history, e.g. referring to a previous topic (e.g., Table 1, Turn 11).
GENERIC	This RG has a prompt that is not conditioned on the conversation so far (e.g., Table 1, Turn 7).

Table 3: Prompt Priorities

This system allows the user to initiate topics (e.g. the bot starts talking about cats if the user utterance is *i want to talk about cats*), allows RGs to initiate topics (see Table 1, Turn 4), allows multiple RGs to talk seamlessly about the same topic (see Table 1, Turn 10), and allows RGs to signal when a topic should be finished (see Table 1, Turn 7).

3.3 Response Priority Ranking System

We use a priority system to decide which response generator's response should be selected on each turn. When generating responses, each RG provides one of the **response priorities** in Table 2.[4] This hierarchy supports the ability to preserve conversational continuity (*STRONG_CONTINUE*), while remaining responsive to the user's initiative (*FORCE_START*). Though it is a relatively simple rule-based system, we have found it well-suited to our needs. The priority levels are clear to understand, and make it easy to modify behavior. By avoiding a centralized response-choosing module, our design allows RGs to decide themselves whether or not they should respond, and whether their response is high quality. This makes it easier for multiple people to work on different RGs, each with self-contained logic. Lastly, if one RG encounters an error, timeout, or inability to find relevant content, the other RGs provide alternatives.

3.4 Response-and-Prompt System

As described in Section 2, on some turns the bot utterance consists of a **response** from one RG, followed by a **prompt** from another RG. This system is useful when the responding RG can handle the user's current utterance, but is unable to take the conversation forward (see Table 1, Turn 10) or when the responding RG has finished talking about one topic, and another RG is needed to supply a change of topic (see Table 1, Turn 7). The response-and-prompt system makes it easy to always supply the user with a strong path forward in the conversation (e.g. by asking the user a question).

3.5 Prompt Priority Sampling System

While we use a deterministic ranking system to choose the highest-priority response (Section 3.3), *prompts* often represent changes of topic, which are less restricted by context, and (in human-human conversations) tend to have a degree of randomness. Thus, we use a priority *sampling* system to select a prompt. When generating prompts, each RG supplies one of the **prompt priorities** in Table 3.

Under the Priority Sampling module, if a *FORCE_START* prompt is supplied, we choose it. Otherwise, we sample from a manually-specified distribution over the remaining priorities, masking out any that

[4] In case of a tie, we tie-break using a manually-specified priority ordering of the RGs.

Training Regime	# MIDAS Training Set	Chirpy Training Set # Silver	Chirpy Training Set # Gold	Chirpy Test Set Micro-F1
MIDAS (baseline)	10,090	0	0	0.53
MIDAS+self-training ($\tau = 0.95$)	10,090	41,152	0	0.54
MIDAS+self-training ($\tau = 0.75$)	10,090	62,150	0	0.54
MIDAS+supervised	10,090	0	2,407	**0.81**

Table 4: Performance of our Dialogue Act model under different training regimes.

are not present on this turn. The distribution is biased towards maintaining continuity of discussion (*CURRENT_TOPIC* $\gg$ *CONTEXTUAL* $>$ *GENERIC*). Then, among the RGs that produced a prompt of the sampled priority, we sample one prompt, using a manually specified distribution over the RGs. This system allows us to specify scripted transitions when desired, and to provide variety via randomness, while still enabling us to tune the likelihood of changing topic, which is an important controllable parameter in chit-chat conversations (See et al., 2019).

4 NLP Pipeline

The NLP Pipeline is run at the start of every turn (see Figure 1), and contains modules that annotate the user's utterance with information that is useful for other parts of the bot.

4.1 CoreNLP

On each turn of the conversation, we annotate the the user's utterance using the Stanford CoreNLP toolkit (Manning et al., 2014), which runs on a remote EC2 module with CPU only. We use the following CoreNLP annotators: tokenization, sentence splitting, part-of-speech tagging, lemmatization, named entity recognition, constituency parsing, dependency parsing, coreference resolution, and sentiment analysis. Due to the format of the user utterances (lowercase with no punctuation), we use the caseless models[5] for part-of-speech tagging, constituency parsing and named entity recognition.

4.2 Dialogue Act Classifier

Dialogue acts can support understanding of user intent (Stolcke et al., 2000), and have been successfully employed in previous Alexa Prize socialbots (Yu et al., 2019). To build a dialogue act classifier, we finetuned the HuggingFace implementation (Wolf et al., 2019a) of a BERT-based classification model (Devlin et al., 2018) on the MIDAS dataset (Yu and Yu, 2019). The dataset contains 12,894 examples, where each example is a bot utterance,[6] the user's response to that utterance, and the user's dialogue act.[7] The dataset was collected by Gunrock (Yu et al., 2019), the winner of the 2018 Alexa Prize competition. Unlike other dialogue act datasets, such as SWBD-DAMSL (Jurafsky et al., 1997), which are designed for human-human dialogue, the MIDAS annotation schema was specifically designed for human-chatbot dialogue.

Though this baseline model achieved a micro-average F1-score of 0.78 on the MIDAS test set, we wished to evaluate its performance in our *own* bot's conversational setting. We hand-labeled a 'Chirpy' test set containing 602 examples from our bot's conversations. The same baseline model achieved only 0.53 on this test set (see Table 4). We suspect the performance drop is due to the distributional difference between the utterances generated by our bot and by Gunrock. To improve performance on our data, we experimented with self-training (McClosky et al., 2006). Using the baseline model, we labeled a large number of unlabeled examples from our own bot's conversations. Examples whose label was predicted with a confidence score greater than a threshold τ were added to our training set. Using $\tau = 0.75$ and $\tau = 0.95$ added 62,150 and 42,152 silver-labeled training examples, respectively. After training on these expanded datasets, we re-evaluated on our own test set. The inclusion of

[5] `https://stanfordnlp.github.io/CoreNLP/caseless.html`

[6] The bot utterance is included because it contains context essential to understand the user utterance (Yu and Yu, 2019). For instance, the user utterance 'tiger king' is an *opinion* when in response to 'What is the best show?' and a *statement* when in response to 'What is the last show you watched?'.

[7] To better fit our needs, we modified the label space as described in Section C.1.

the silver-labeled data did not substantially boost performance (see Table 4). Finally, we turned to supervised training, and hand-labeled an additional 2,407 examples from our own bot's conversations (procedure described in Section C.2). After training on the MIDAS data and this data, we achieved a much higher micro-F1 of 0.81 on the Chirpy test set.

In our bot, we run the Dialogue Act classifier on an EC2 machine with one NVIDIA T4 Tensor Core GPU, annotating every user utterance in the conversation. We find that its accuracy is best on classes with low variance in user utterances, such as *positive answer*, while classes with high variance, such as *statement*, are more difficult. However, even for the low variance classes, the classifier's labels are very useful – we are able to achieve much higher recall in recognizing *positive answer* and *negative answer* by using the classifier's labels, compared to regexes or word lists.

4.3 Question Classifier

Users often spontaneously ask factual questions, personal questions, follow-up questions, and even questions unrelated to the current topic. Recognizing and answering these questions is important, particularly for user initiative, but is also non-trivial, as user utterances do not contain punctuation.

To recognize questions, we initially used the Dialogue Act classifier's labels (which include question types like *factual question* and *open-ended question*). However, this did not work well; the classifier seemed to condition too much on the bot utterance preceding the user utterance – which is less useful for recognizing questions than other dialogue acts. Instead, we fine-tuned a RoBERTa model (Liu et al., 2019; Wolf et al., 2019a) on an simplified version of the Dialogue Act training data, framing the task as binary classification, conditioned only on the user utterance. This model achieved an F1-score of 0.92 and improved the reliability of question detection.

The classifier's labels are used to determine when certain RGs should respond – for example, when the Evi RG (Section A.3) should answer a factual question. The labels are also useful for the neural generative models (Sections 5.2, 5.3, 5.7). We observe that the GPT-2-based models are much more likely to answer (rather than ignore) a user's question if a question mark is present. Thus, we use the classifier labels to determine when to append a question mark to the user utterance.

4.4 Entity Linker

A key part of our high-coverage strategy (Section 1) is *entity linking* – detecting when the user is referring to an entity, and identifying the correct entity. To obtain our pool of potential entities, we processed a dump[8] of English language Wikipedia. For each article (i.e. each entity E), we collected (a) the *pageview* (number of views in one month), and (b) the *anchortext distribution* $P_{\text{anchortext}}(a|E)$.

To compute the anchortext distribution for an entity E, we count the number of *anchortexts* (i.e., strings, lowercased) that are used as hyperlinks to E across Wikipedia (e.g., the entity Barack Obama may be referred to using the anchortexts *barack obama*, *obama*, or *president obama*). Then:

$$P_{\text{anchortext}}(a|E) = \frac{\text{count(links from } a \text{ to } E)}{\sum_{a' \in A(E)} \text{count(links from } a' \text{ to } E)} \tag{1}$$

where $A(E)$ is the set of all anchortexts that link to E. We store each entity, along with its Wikipedia article, pageview, anchortext distribution, and Wikidata categories[9] in an ElasticSearch index.

After we receive the user's utterance u, we assemble the set of candidate spans S. S contains all n-grams in u with $n \leq 5$, excluding n-grams that consist only of stopwords. We then query ElasticSearch to fetch all entities E which have at least one span $s \in S$ among its anchortexts. To determine which entities the user is referring to, we wish to estimate $P(E|s)$, the likelihood that a span s is referring to an entity E. We model $P(E|s)$ as a Bayesian system:

$$P(E|s) \propto P(E) \times P(s|E). \tag{2}$$

We assume that $P(E)$ is proportional to the pageview for the entity E, and $P(s|E) = P_{\text{anchortext}}(s|E)$. Therefore, we define the score(s, E) of a span s and and entity E to be:

$$\text{score}(s, E) = \text{pageview}(E) \times P_{\text{anchortext}}(s|E). \tag{3}$$

[8] `https://dumps.wikimedia.org`

[9] For each entity, we collected all its ancestors via the *instance of* and *subclass of* relations. For people entities, we also used the *occupation* relation.

The output of the entity linker is a priority-ordered list of (s, E) pairs. The ordering is calculated using manually-curated rules and thresholds on the following features: (a) the score of (s, E), (b) the maximum unigram frequency[10] of s, (d) whether E is in a Wikidata category that is expected for this turn[11], (c) whether s is contained inside any other linked span (priority is usually given to the larger span). The output of the entity linker is primarily used by the entity tracker (Section 3.2) to identify the current entity under discussion.

Limitations We found the entity linker to be one of the hardest components of our bot to build. One difficulty is that our notion of an entity – anything with a Wikipedia article (e.g. *Cat* or *Musical instrument* in Table 1) – is much broader than the traditional definition of Named Entities (which is typically restricted to particular types, such as people and locations). Our motivation in this definition was to enable high-coverage world knowledge by enabling any Wikipedia article to become a focus of discussion. However, this made the entity linker's job much more difficult. The need to detect an extremely broad range of entities, with no restriction to certain types, made it much more difficult to find a good precision/recall tradeoff, leading to both false positive and false negative problems in the bot. In the future, we will need to develop better approaches for identifying our expanded notion of entities, or find a way to support high coverage of topics without relying as much on the entity linker.

ASR Error Robustness As we do not have access to original user audio, ASR errors are a major source of difficulty, particularly when they occur within entity names. For example, if the user wants to talk about the film *Ford v Ferrari*, but the ASR transcription is *four v ferrari*, our entity linker will fail to identify the correct entity, as the span *four v ferrari* is not among the anchortexts for the entity *Ford v Ferarri*. To address this, we adapted our entity linker to be robust to phonetically-similar spans and anchortexts; our method is similar to Chen et al. (2018).

First, we converted all Wikipedia entity anchortexts to their phoneme and metaphone representations (e.g., *Harry Potter* to `'HH EH R IY P AA T ER'` and `'HRPTR'`) with a grapheme-to-phoneme tool[12] and the double metaphone algorithm,[13] and indexed the mapping from anchortext phonemes to Wikipedia entities in ElasticSearch. When running the entity linker, we convert all spans $s \in S$ to their phonetic representations and query the ElasticSearch index, which returns a set of anchortexts A_{phon} that have similar phonetic representations to any of the spans queried. This allows us to expand the candidate pool for each span s, from entities for which s *is* an anchortext, to entities for which s is *phonetically similar* to an anchortext. Finally, we redefine $P(s|E)$ as follows: for each anchortext $a \in A_{\text{phon}}$, we start by finding its best-matching span $s^*(a) = \arg\max_{s \in S} \text{sim}(s, a)$ where $\text{sim}(\cdot, \cdot)$ is a phoneme similarity function[14] between 0 and 1; then, we filter out anchortexts that are phonetically too dissimilar to each span with a threshold of 0.8, resulting in a set of anchortexts for each span $A(s) = \{a | a \in A_{\text{phon}}, s = s^*(a), \text{sim}(a, s) \geq 0.8\}$. Finally:

$$P(s|E) \propto \begin{cases} \max_{a \in A(s)} \text{count(links from } a \text{ to } E) \times \text{sim}(s, a) & A(s) \neq \emptyset \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

This definition of $P(s|E)$ replaces $P_{\text{anchortext}}(s|E)$ in Equation (3).

5 Response Generators

In this section, we describe our Response Generators (RGs). Additional minor RGs are described in Appendix A. We also describe *treelets* (Section 5.1), a system we used to organize many of our RGs.

5.1 Treelets: A System to Organize Dialogue Graphs

Many of our response generators rely on *treelets*, a modular programming abstraction which represents a single node in a dialogue graph. The treelet system is largely based on dialogue trees (Weizenbaum et al., 1966) and dialogue-frame-based systems such as GUS (Bobrow et al., 1977). We define a treelet to be a small, 1-turn dialogue 'tree' that manages all decisions necessary to produce a bot

[10] The maximum unigram frequency of s is the frequency of the most common unigram inside s, computed using this unigram frequency list for spoken English: `http://ucrel.lancs.ac.uk/bncfreq/flists.html`

[11] For example, if the bot asked *What's your favorite movie?*, an expected Wikidata category is *film*.

[12] `https://pypi.org/project/g2p-en/`

[13] `https://pypi.org/project/metaphone/`

[14] implemented on lists of phonemes with Python's `difflib.SequenceMatcher`

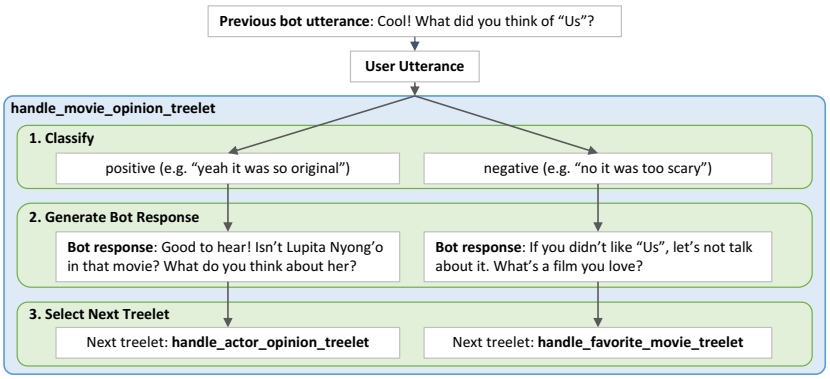


Figure 2: An example *treelet* for the Movies RG.

response given a user's utterance. This involves interpreting the user utterance, creating the bot's response, and specifying the treelet that should take control on the next turn.

Typically, a treelet performs three actions: (1) it classifies the user's utterance into one of several branches, (2) it produces an appropriate bot response for that branch, (3) it specifies the next treelet. Treelets throughout our bot may classify user utterances by using regexes, outputs from our NLP pipeline (the dialogue act classifier is frequently used for this purpose), or changes in entity (e.g., if a treelet in the Movies RG detects that the current entity has changed to "food" after the user says "let's talk about food", the current Movies treelet may select a branch that returns no response). Bot responses may be handwritten or dynamically generated (we use both throughout our system). An example from the Movies RG is shown in Figure 2.

Like dialogue trees in general, treelets provide a well-controlled, predictable and easily interpretable conversation flow. From an engineering and implementation perspective, treelets have several advantages, such as allowing modular organization of code and dialogue, easily enabling cycles when desired (by having treelets point to each other with repeats or loops), and minimizing code duplication by allowing many treelets to point to the same successor.

5.2 Neural Chat

The Neural Chat RG's goal is to empathetically discuss personal experiences and emotions with the user, using responses generated by a GPT-2-medium (Radford et al., 2019) model finetuned on the EmpatheticDialogues dataset (Rashkin et al., 2019). The dataset consists of conversations between a *speaker*, who describes an emotional personal experience, and a *listener*, who responds empathetically to the speaker's story. Our model is trained in the listener role.

The Neural Chat RG has 7 discussion areas: current and recent activities, future activities, general activities, emotions, family members, living situation, and food. A discussion begins by asking the user a **starter question** (e.g, *What do you like to do to relax?* for the 'general activities' area). Some starter questions are conditioned on the time of day (e.g. *What did you have for breakfast/lunch/dinner today?* for the 'food' area). Starter questions can be asked as part of the launch sequence (Table 1, Turns 2 and 3), as generic changes of topic, (*Do you have any plans for the weekend?*), or can be triggered contextually (*You mentioned your boyfriend. How did you guys meet?*). On each subsequent turn of the discussion, we generate 20 possible responses from the GPT-2 model using top-p sampling with $p = 0.9$ and temperature 0.7. To provide a strong path forwards in the conversation, we generally choose a GPT-2 response containing a question. However, if under a third of the sampled responses contain questions, we interpret this as an indication that the model is not confident in asking a question on this turn. In this case, we choose a non-question and end the Neural Chat discussion. Under this strategy, each Neural Chat discussion contains 2.75 bot utterances on average.

The model was finetuned using the HuggingFace ConvAI code[15] (Wolf et al., 2019b) and is hosted on a GPU-enabled EC2 machine with one NVIDIA T4 Tensor Core GPU. To keep latency low we

[15] `https://github.com/huggingface/transfer-learning-conv-ai`

Strategy	Preamble
NO_SHARE	I wanted to check in with you.
POS_OTHERS	I've noticed that a lot of people are feeling pretty positive today!
POS_BOT	I wanted to say that I'm feeling pretty positive today!
POS_BOT_STORY	**POS_BOT** + I just went for a walk outside, and it felt great to get some fresh air.
NEG_OTHERS	I've noticed that a lot of people are feeling kind of down recently.
NEG_BOT	I wanted to say that I've been feeling kind of down recently.
NEG_BOT_STORY	**NEG_BOT** + I've been missing my friends a lot and finding it hard to focus.
NEGOPT_OTHERS	**NEG_OTHERS** + But I think its important to remember that things will get better.
NEGOPT_BOT	**NEG_BOT** + But I think its important to remember that things will get better.
NEGOPT_BOT_STORY	**NEGOPT_BOT** + Just earlier today I took a walk outside and the fresh air helped me get some perspective.

Figure 3: Strategies for the emotion-focused Neural Chat starter question. **POS/NEG/NEGOPT** refer to positive/negative/negative+optimistic emotion. **OTHERS/BOT** refer to whether the emotion is attributed to other people, or to the bot. **STORY** indicates that the bot shares a personal anecdote.

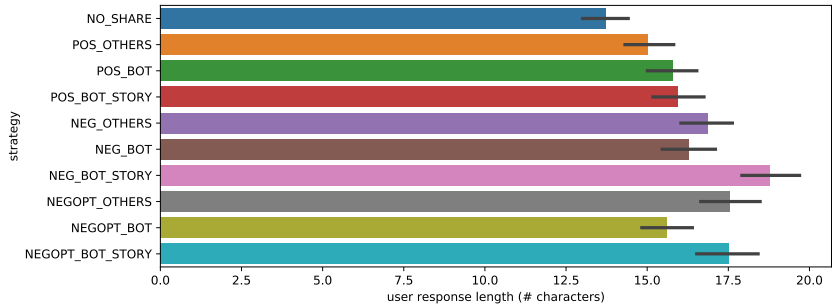


Figure 4: Effect of Neural Chat emotion-focused starter question strategies on user response length.

truncate the conversational history supplied to the model, so that the total number of GPT-2 tokens is below 800. Given that neural models have been shown to make poor use of longer conversational history (Sankar et al., 2019), this truncation does not seem to be a limiting problem currently.

Emotion-focused Conversations As part of our goal to provide an emotionally-engaging experience (Section 1), we would like to give users space to share their genuine feelings, then respond empathetically to them. This is especially important during the Coronavirus pandemic (Section A.1), which is an emotionally challenging time for many. Given our basic starter question *I hope you don't mind me asking, how are you feeling?*, we tried several different preambles to precede the question (Table 3). Figure 4 shows the effect of the different strategies on the length of the user's response. We find that the basic **NO_SHARE** strategy has the shortest average response length, indicating that the bot's emotional observations (whether about the bot or about other people) lead users to give more substantive responses. Users tend to give longer responses when the bot expresses negative emotions (**NEG** and **NEGOPT**) than positive (**POS**) – this may be because acknowledging negative emotions makes users feel more comfortable to answer the question honestly, rather than superficially (e.g. *i'm fine*). Furthermore, adding a personal anecdote (**STORY**) to the negative bot emotions led to longer responses – users may have responded more because the bot was more specific or relatable. For positive emotions (**POS**), users are more responsive when the bot attributes the positive emotion to itself (**BOT**), than to other people (**OTHERS**). However, for negative emotions (**NEG** and **NEGOPT**), the opposite is true. We also experimented with including the user's name in the starter question, but found that this made no difference to user response length.

Discussion Our neural generative model has several recurring weaknesses which impact overall user experience. First, it frequently asks for already-provided information, asks nonsequitur questions, makes unfounded assumptions about the user, and confuses its own previous responses with the user's. This demonstrates that incorporating commonsense reasoning is a priority in neural generation. Second, while the model generally produces interesting and relevant responses to longer user utterances, it performs poorly when the user utterance is short or low-content (e.g. *okay*, *i don't know*, *nothing*) – probably because these utterances are unlike the much longer and contentful EmpatheticDialogues

training data. The model tends to respond to these with bland responses that further fail to drive the conversation to any interesting substance. This problem with short user responses is one reason why we focused on finding starter questions that lead to substantial user responses (Figure 4).

Due to these difficulties, most conversations with the GPT-2 model tend to fall apart after a few turns, as the bot will eventually ask a question that doesn't make sense, which will flummox the user. This is one reason why we designed the Neural Chat module around shorter sub-conversations. However, overall, we are excited that neural generation is now able to interact successfully with real people, within certain constraints (such as keeping the discussion short, bookending it between handwritten starter questions and wrapup phrases, and providing a strong path forward through questions).

5.3 Wiki

To support our goal of high-coverage world knowledge (Section 1), the Wiki RG uses Wikipedia articles as grounding to discuss any entity that interests the user. Our goal is to allow the user to conversationally discover interesting information about the entity.

Data To prepare the Wikipedia data, we downloaded the most recent Wikipedia dump,[16] processed it using MWParserFromHell[17] and Spark,[18] and uploaded it into an ElasticSearch index. The Wiki RG can then query the ElasticSearch index to obtain the Wikipedia article for an entity.

Behavior On each turn, if it's not already active, the Wiki RG can start to talk about the current entity (Section 3.2) by asking the user an **open ended question**, such as *What do you find interesting about it?*. If the entity is in one of 25 commonly-encountered types (determined using Wikidata categories), such as books or foods, we use a more specific question, such as *What did you think of BOOK_ENTITY's story?* or *I love trying out new flavor combinations. What do you like to have FOOD_ENTITY with?*. These questions are designed to elicit contentful user responses, which can be matched to specific sentences in the Wikipedia article using TF-IDF overlap. The RG also offers interesting facts (i.e. 'TILs') scraped from the /r/todayilearned subreddit, if available. If we have given enough TILs or we have no TIL left to offer, we will start suggesting sections of the Wikipedia article to the user. A short example Wiki interaction is shown in Turns 11-13 of Table 1.

Conversational Styling We use this RG as a testbed for our conversational paraphrasing system. The system takes as input the truncated conversational history, and some knowledge context (either a TIL about the current entity, or an excerpt of the Wikipedia article, selected based on TF-IDF similarity to the user's response to an open-ended question). It outputs a conversational-sounding paraphrase of the knowledge context. The model was trained by finetuning a GPT-2-medium language model (Radford et al., 2019) on a processed and filtered version of the TopicalChat dataset (Gopalakrishnan et al., 2019). The paraphrases are generated using top-p decoding with $p = 0.75$ and temperature $\tau = 0.9$, and we pick the one which has the highest unigram overlap with the knowledge context.

Challenges One major challenge while performing conversational styling is that the model sometimes produces **factually incorrect** or nonsensical conversational paraphrases. Another challenge is that integrating the paraphrasing model with the rest of the system requires **explicit directives** such as "continue talking about same knowledge piece", "pick another fact", "change entity" which the model currently does not produce. For instance, sometimes the generated paraphrase just asks a question or mentions an incomplete piece of information, with the expectation of completing it in the next turn. Currently we apply some heuristics such as presence of *Did you know ... ?* style questions or low unigram overlap to determine that the same snippet needs to be paraphrased again.

More broadly, there are challenges around **interestingness of content**. The majority of content on Wikipedia isn't very interesting and social. While the TILs remedy that to some extent, finding interesting parts of raw text is still an open question and quite important in the open-domain conversational setting. Another major challenge is **content selection and discoverability**. The user doesn't know the extent of the knowledge that our system possesses for an entity. In a visual interface, the user can scroll through the article or look at a table of contents. While we partly remedy this by suggesting section titles to illustrate the kind of content we can talk about, a better system could

[16] `https://dumps.wikimedia.org/backup-index.html`

[17] `https://mwparserfromhell.readthedocs.io/en/latest`

[18] `https://spark.apache.org`

Policy Name	Continuation Rate	CI
CONVINCED_AGREE	0.526829	0.0348712
ALWAYS_AGREE	0.586638	0.0086009
LISTEN_FIRST_DISAGREE	0.587045	0.0127898

Table 5: Continuation rate for each agreement policy. The Confidence Intervals (CI) differ due to different sample sizes (ALWAYS_AGREE receives 0.5 of traffic, LISTEN_FIRST_DISAGREE receives 0.3, CONVINCED_AGREE receives 0.2).

perhaps understand what different parts of a Wikipedia article are talking about, and steer conversation in that direction.

5.4 Opinion

Exchanging opinions is a core part of social chit-chat. To form a stronger sense of personality, and to seem more relatable, it is important that our bot can also express its opinions. The Opinion RG's goal is to listen to users' opinions on certain topics, and reciprocate with its 'own' opinions (sourced from Twitter) on those topics.

Data To collect both positive and negative opinions, we queried a Twitter stream[19] using a regex to collect tweets of the form `'i (love|like|admire|adore|hate|don't like|dislike) TOPIC because REASON'`, where `TOPIC` and `REASON` can be any text. We collected 900,000 tweets, which are stored on a Postgres table hosted on AWS Relational Database Service (RDS). Of these, we manually whitelisted 1012 reasons across 109 popular topics. To avoid speaking inappropriately about sensitive topics, we only whitelist uncontroversial entities (such as animals, foods, books/movies/games, everyday experiences such as working from home, being sick, days of the week, etc.), and ensured that all reasons, including negative ones, are inoffensive and good-spirited.

Behavior Currently, the Opinion RG activates when the user mentions one of the whitelisted entities (e.g. Table 1, Turn 8). We ask whether the user likes the entity and classify their response using the CoreNLP sentiment classifier (Section 4.1). We then either agree or disagree with the user. If we disagree, we either ask the user for their reason for their opinion, or supply a reason why we disagree, and ask what they think of our reason. Ultimately, we want the user to have a positive experience with our bot, so regardless of whether we disagree or agree with the user, we will ask the user their opinion on a related entity, and always agree with the user about the new entity. The conversation may end earlier, as we detect on each turn whether the user is still interested via their utterance length. If the utterance contains less than 4 words, and it does not contain any of the 'agreement' words (such as 'same', 'me too', etc.) we will hand off the conversation to another RG. Even when the RG is not active, it keeps track of whether the user has already expressed an opinion on an entity, by applying a regex similar to that applied to the tweets.

Agreement Policies Disagreement is an unavoidable part of human-human conversations, and we hypothesize that occasional disagreement is necessary in order for our bot to have a convincing and individual personality. To test this, we implemented three policies (full details in Appendix F): (i) `ALWAYS_AGREE` – we always agree with the user's sentiment on the entity; (ii) `LISTEN_FIRST_DISAGREE` – first we ask the user's reason for liking/disliking the entity, then we offer our reason for disagreeing with their sentiment; and (iii) `CONVINCED_AGREE` – we initially disagree with the user's sentiment on the entity, but after the user gives their reason for liking/disliking the entity, we switch our sentiment to match the user's (i.e. we are convinced by the user). To evaluate the policies, we ask the user *Would you like to continue sharing opinions?* and interpret the desire to continue is an indication of a successful policy. Table 5 shows that users prefer `ALWAYS_AGREE` and `LISTEN_FIRST_DISAGREE` over `CONVINCED_AGREE`, and all policies have high continuation rates, suggesting that disagreement can be a positive and stimulating part of a conversation, but that the manner and delivery of the disagreement is an important factor.

[19] https://developer.twitter.com/en/docs/tutorials/consuming-streaming-data

5.5 Movies

The Movies RG is designed to deliver a high-quality scripted conversation about a movie the user specifies, using information drawn from the Alexa Knowledge Graph.[20] Currently, the RG is activated when the user asks to talk about movies, mentions a movie keyword (such as *movies* or *film*) or talks about any movie-related entity (e.g. *Saving Private Ryan*, *Meryl Streep*, *the Coen brothers*, etc.). Once activated, the RG typically asks the user to name a movie, asks the user's opinion on it, gives a fun fact about the movie, asks the user their opinion on an actor in the movie, then asks the user if they've seen a different movie featuring that actor (See Turns 4-7 in Table 1). The RG uses treelets (Section 5.1) to organize the dialogue graph, hand-written templates to form the bot utterances, and a mixture of regexes and the CoreNLP sentiment classifier (Section 4.1) to classify the user's responses.

The primary weakness of this RG is that, as a scripted dialogue graph, it does not offer very high user initiative (one of our design goals – Section 1). However, this RG was important especially early in the competition when our more flexible RGs were still under development, and we needed more content. Another difficulty we faced was the latency of the Alexa Knowledge Graph, which was sufficiently slow that we were limited to one query per turn; this limited the scope of interesting information that we could pull about an entity and heavily influenced the design of our dialogue tree.

5.6 Music

Similar to the Movies RG, the Music RG is designed to deliver scripted conversations about musical entities that the user specify. The RG is activated when a musician/band or a music keyword (such as *music* or *songs*) is mentioned. Once activated, the Music RG engages in a conversation specific to the type of the musical entity that was mentioned. Unlike the Movies RG, the Music RG has a randomized internal prompting system that allows the conversation to be centered around music even when a scripted conversation is exhausted for a specific entity. For example, after the Music RG goes until the end of a scripted conversation for a musician, it can ask for an internal prompt, and start a conversation about musical instruments, songs, or music in general. The randomized nature of the internal prompting system makes the conversation more flexible, and mitigates some of the weaknesses of scripted conversations mentioned in Section 5.5.

5.7 Neural Fallback

Our Fallback RG's responses – e.g., *Sorry, I'm not sure how to answer that* (Section A.3) – are a poor user experience, making the user feel ignored and not understood. The Neural Fallback RG aims to generate a better fallback response using our GPT-2 EmpatheticDialogues model (Section 5.2) – to be used only if every other RG (excluding Fallback) has no response. If the neural fallback response is chosen, another RG immediately produces a prompt to move the conversation in another direction. After some filtering (e.g. removing responses that ask questions or give advice), the neural fallbacks can work well as a way to better acknowledge and show understanding of what the user said, such as on Turn 11 of Table 1. A remaining issue is latency – generating from the GPT-2 model is typically the slowest component in the turn, which is a poor tradeoff if we don't use the neural fallback.

5.8 Categories

The Categories RG was originally designed to ask handwritten questions about certain categories; for example, *Where's a place you would love to visit?* for the 'travel' category. These questions may be asked when the current topic is 'travel', or used as generic changes of topic (Table 1, Turn 7). The goal is for the user to name an entity (e.g. *japan*) that can form the basis for an interesting discussion (e.g. with the Wiki or Opinion RGs). However, we found that repeatedly asking users to think of entities led to decision fatigue, with many users failing to think of an entity.[21] As alternatives to the `QUESTION` strategy, we experimented with two other strategies: `STATEMENT`, in which the bot just makes an observation about a relevant entity (e.g. *Mexico is one of my favorite places. I love the food and beaches!*), and `STATEMENT+QUESTION`, which combines the other two strategies. Table 6 shows that the statement followed by a question elicited the most new entities. This may be

[20]The Alexa Knowledge Graph is an Amazon-internal resource; our team was given access to parts of it.

[21]If the user does not name a new entity, we respond either with a handwritten acknowledgment and new question (if the user said *I don't know* or similar), or with the GPT-2 model (Section 5.7).

Strategy	Proportion of Turns with New User Entities	CI
STATEMENT	0.272	0.012
QUESTION	0.264	0.027
STATEMENT+QUESTION	0.328	0.016

Table 6: Rate at which users suggest new entities, for different strategies in the Categories RG. The entities are extracted using our Entity Linker (see Section 4.4). (CI: Confidence Interval)

Strategy	Re-offense Rate	Confidence Interval
WHY	0.520	±0.049
WHY+NAME	*0.638*	*±0.07*
AVOIDANCE	0.554	±0.049
AVOIDANCE+NAME	0.391	±0.061
AVOIDANCE+PROMPT	0.583	±0.047
AVOIDANCE+NAME+PROMPT	**0.346**	**±0.066**
COUNTER+PROMPT	0.567	±0.042
EMPATHETIC+PROMPT	0.461	±0.046

Table 7: Re-offense rates for different response strategies to offensive utterances. Italic and bold denote the worst and best performing, respectively.

because the statement gives users an example, and takes the focus off the user for a moment, before prompting them with a question. This is a more natural, mixed-initiative experience than simply asking a question.

5.9 Offensive User

Users sometimes give offensive or critical utterances, and it is important for our bot to handle these appropriately (Curry and Rieser, 2018, 2019). Unsurprisingly, there is an inverse relationship between the presence of offensive user utterances in a conversation and the conversation rating (Figure 9). Our goal is to redirect the user away from making offensive comments, towards topics the bot can discuss.

On each turn, the Offensive User RG checks the user's utterance for offensive language using a blacklist of offensive phrases.[22] If the user's utterance is more critical than offensive, we respond with an apologetic strategy (see Turn 10 of Table 1). For offensive user utterances, we implemented two immediate response strategies: asking the user why they made the offensive remark (WHY); or politely avoiding the topic (AVOIDANCE). In addition, for AVOIDANCE, we experimented immediately changing the topic by using a prompt in the same turn (AVOIDANCE+PROMPT). For each of these configurations, we experimented with mentioning the user's name (NAME), or not. We also implemented the strategy COUNTER+PROMPT, inspired by Brahnam (2005), which directly confronts the user before changing topic, and EMPATHETIC+PROMPT, inspired by Chin et al. (2020), which empathizes with the user before changing topic. The full details can be found in Appendix E.

Table 7 shows the effect of each strategy on re-offense rate (i.e., the probability that the user says another offensive utterance in the same conversation). We find that mentioning the user's name reduces the likelihood of re-offense when we use the avoidance strategy, but increases re-offense rate when we ask the user why they made an offensive remark. We hypothesize that by using their name, we motivate the user to defend themselves, which prolongs the offensive conversation. We find that our AVOIDANCE+NAME+PROMPT method outperforms the empathetic method (EMPATHETIC+PROMPT) and the confrontation method (COUNTER+PROMPT).

[22] https://www.freewebheaders.com/full-list-of-bad-words-banned-by-google/. Our offensive classifier is also used by our RGs to check that externally-sourced content (e.g. news articles, Wikipedia articles, fun facts) are inoffensive.

6 Analysis

6.1 Relationship between Rating and Engagement

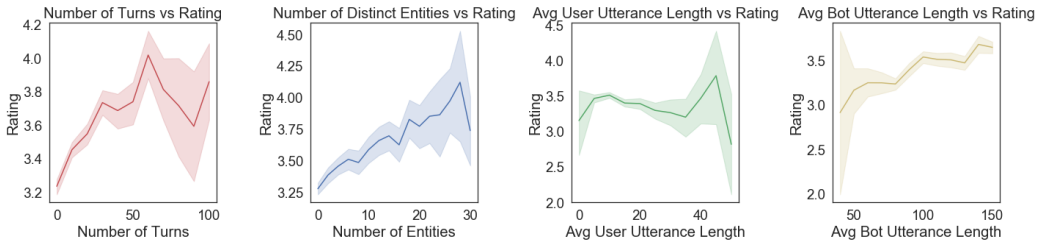


Figure 5: Engagement metrics vs rating

We measured four metrics of engagement: number of turns in the conversation, number of distinct entities discussed during the conversation, average length of the user's utterances, and average length of the bot's utterances. Figure 5 shows that rating increases with number of turns and number of entities, but ultimately drops off. In an analysis of Alexa Prize bots, Venkatesh et al. (2018) found that across all bots, conversation length was positively correlated with rating; however, one possible explanation for our result is that our bot has limited content and at some point, the users become dissatisfied as their experience is no longer novel.

In an analysis of the NeurIPS ConvAI2 challenge, Dinan et al. (2019) found a positive relationship between user utterance length and rating. We expected a similar result, thinking more talkative users would be more actively engaged. However, Figure 5 shows that rating increases with user utterance length until about 12 characters, and then decreases. Since many of our bot's questions encourage short answers (e.g. *What's your favorite animal?*; *Would you like to talk about science?*), and it is generally more difficult for our bot to correctly understand and handle longer answers,[23] users who give longer answers may have a worse experience. For this reason, the result shown may reflect the limitations of our bot, more than a user preference for giving shorter responses.

Average bot utterance length is positively correlated with average rating, with high variance in rating for shorter bot utterances. A confounding factor is that different response generators have varying average response lengths and relationship with user experience (Section 6.4) – e.g., the Offensive User RG tends to give short responses, and has a negative relationship with ratings. Response generators giving longer responses tend to have positive or neutral relationships with rating. Therefore, this plot may more reflect the UX of our response generators than a user preference for longer responses. These results may also reflect the inherent noise in user Likert-scale ratings (Liang et al., 2020).

6.2 Relationship between Rating and User Dialogue Acts

To understand how users' dialogue acts relate to our bot's performance, we applied a regression analysis, using the statsmodels Seabold and Perktold (2010) implementation of Ordinary Least Squares, to the distinct dialogue act classifier labels for all utterances of a conversation and the ultimate rating of that conversation. These results are shown in Table 7. As we would expect, *appreciation* is associated with higher ratings and *complaint* with lower ratings.

One of our design goals was having mixed-initiative dialogue. In general, dialogue acts associated with low user initiative, such as *comment*, *pos_answer*, *statement*, and *back-channeling* were more positively associated with rating than dialogue acts associated with high user initiative, such as *command*, *open_question_opinion*, and *open_question_factual*. A possible explanation for this is that users take more initiative when dissatisfied with the current conversational direction, for example by giving a command to change the topic. On the other hand, users giving yes-answers or back-channeling, are likely being compliant with the bot's direction, which may reflect greater overall satisfaction. It is possible that these results are more indicative of user satisfaction with our content than of a user preference for low vs high initiative.

[23] As an exception, our neural generation models perform *better* on longer user utterances; see Section 5.2.

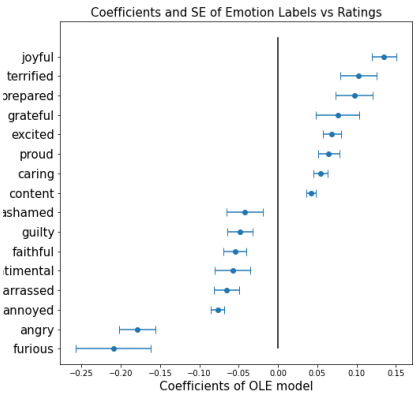


Figure 6: Regression coefficients for Emotion vs Rating

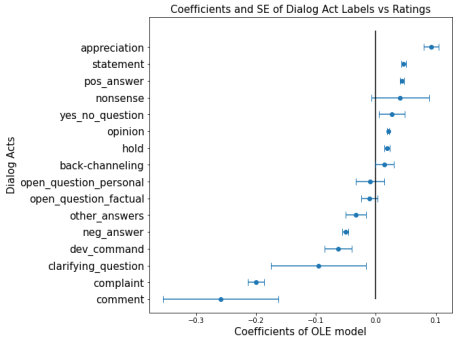


Figure 7: Regression coefficients for Dialogue Act vs Rating

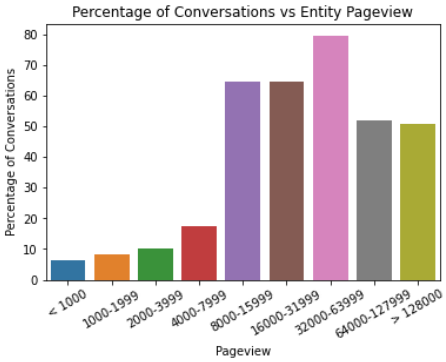


Figure 8: Percentage of conversations in which users initiated discussion of entities with different popularity levels (pageview).

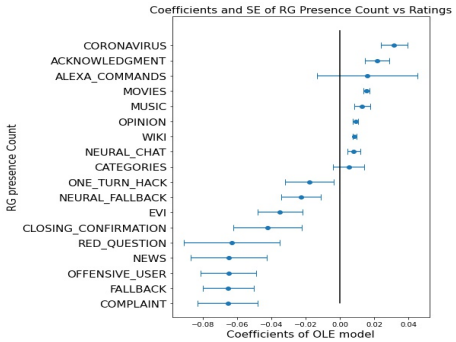


Figure 9: Regression coefficients for Response Generator vs Rating. Launch RG is not included as it is in every conversation.

6.3 Entity Coverage

As part of our design goal to offer high coverage of topics (Section 1), our bot is capable of discussing any Wikipedia entity (Section 3.2), and discussed 7.5 distinct entities on average per conversation. To support user initiative and engage users, we designed our bot to be able to discuss both popular and lesser-known entities. We regard the Wikipedia pageview (Section 4.4) as a measure for an entity's popularity. To measure users' desire to discuss less-common entities, Figure 8 shows the percentage of conversations where users initiated discussion of an entity with different pageview levels. These counts do not include entities initiated by the bot. As the plot shows, a significant number of users wanted to discuss uncommon entities: in 8% of our conversations, users initiated discussion of entities with fewer than 2000 views and 33% of conversations covered at least one entity with fewer than 8000 views. Users who discussed rare entities with the bot appeared to have favorable experiences. Conversations with rare entities (fewer than 16000 pageviews) had an average rating of 3.88, while those without rare entities had an average rating of 3.64.

To understand which entities had the greatest impact on user experience, we used the top 100 most frequent entities as features for a regression analysis, using an Ordinary Least Squares model. Of the 100 most popular entities, 15 had a statistically significant ($p \leq 0.05$) positive impact on rating. These include **animals** ('Cat', 'Dog'), **movies** ('Film', 'Frozen 2', 'Onward (film)'), **food** ('Korean fried chicken', 'Pizza', and 'Ice cream'), and **video games** ('Minecraft', 'Fortnite').

6.4 Effectiveness of Response Generators

We performed a regression analysis on the relationship between response generator use and rating, using the number of turns each RG contributed as features. Figure 9 shows a statistically significant positive relationship between rating and the Coronavirus, Acknowledgment, Movies, Opinion, and Wiki RGs, and a statistically significant negative relationship for Red Question, Complaint, Fallback, Neural Fallback, and Offensive User. The Complaint and Offensive User results may be explained by the fact that users experiencing poor conversations may complain or be offensive, and conversely, some adversarial users deliberately engage negatively and then give poor ratings. A possible cause for the negative Fallback and Neural Fallback results is that these RGs are used when no other RG has a high-quality response, so their use is likely correlated with a worse user experience. As we expected, RGs designed for general conversation had more positive coefficients. Of these RGs, those with more scripted content, i.e. Coronavirus, Acknowledgment, Movies, and Categories had larger positive coefficients than those with less, such as Opinion and Wiki. However, the most significant loss in performance occurs when the bot cannot answer contextually or has an adversarial user.

7 Discussion and Future Work

Full Stack NLP Most NLP research focuses on self-contained tasks. However, an open-domain socialbot, served to a diverse range of customers in widely different contexts, is by no means a self-contained task. Our socialbot is a tapestry of many such components, requiring a deep understanding of each component and how they should work together – a setting we call Full Stack NLP. Often the inputs and outputs of these components are inter-dependent, leading to cascading errors. We made many design choices which delay hard decisions in pipelines, and maximize information exchange between modules. Moving forward, the next avenue for advancing the state-of-the-art would be research on models which perform these tasks jointly and methods which enable training over multiple interdependent tasks with only a small amount of joint supervision.

Domain Shift As a recurring problem, we found that many existing NLP resources didn't work well out-the-box. The main reason for this is that the training data for these resources (typically non-conversational, longform, traditionally-formatted written text) is misaligned with our setting (conversational, shortform, uncased, punctuationless, spoken text). However, a deeper reason is the constantly changing nature of dialogue agents themselves. Even for an extremely related resource (the MIDAS dialogue model, developed for the Alexa Prize, Section 4.2), domain shift was a problem. Recent advances in online- and meta-learning could provide a useful long term solution to this issue.

Conflict and Intimacy Bot-human conversations are fundamentally different to human-human conversations. Users can be adversarial, deliberately testing the bot's boundaries. As socialbot designers, we are eager to avoid a disaster like Microsoft Tay, so we apply strict but overly simplistic methods to block off sensitive topics (Sections 5.4, 5.9). However, this rules out sincere conversation about difficult topics. We observed that users are actually quite resilient to conflict, and can find disagreement stimulating (Section 5.4). We also found that emotional intimacy is reciprocal – users are more inclined to share their feelings after the bot has shared its own (Section 5.2). Going forward, we should continue to take seriously the dangers of speaking inappropriately, but keep in mind the cost – to engagement and to intimacy – of not engaging in difficult topics.

Initiative As part of our goal to support user initiative, we focused on asking users questions to find out which topics interested them. However, this puts pressure on the user to think of a response, especially given the time constraints of Alexa devices. Thus we found that our attempts to let the user take more initiative unfortunately led to decision fatigue. Separately, our ability to support user initiative was limited by our ability to answer followup questions, and to correctly understand long or unexpected user utterances. On balance, we found that asking the user open-ended questions about interesting topics was a good strategy – easier to handle than spontaneous user questions, and less pressuring than asking users to name topics. We see an opportunity for future work to build systems which listen more to the user's knowledge, rather than only providing knowledge.

Acknowledgments

Thank you to Anna Goldie for her advice and guidance to the team. Abigail See's work was supported by an unrestricted gift from Google LLC. We thank Amazon.com, Inc. for a grant partially supporting the work of the rest of the team.

References

Daniel G. Bobrow, Ronald M. Kaplan, Martin Kay, Donald A. Norman, Henry Thompson, and Terry Winograd. 1977. Gus, a frame-driven dialog system. *Artificial Intelligence*, 8(2):155 – 173.

Sheryl Brahnam. 2005. Strategies for handling customer abuse of ECAs. pages 62–67.

Chun-Yen Chen, Dian Yu, Weiming Wen, Yi Mang Yang, Jiaping Zhang, Mingyang Zhou, Kevin Jesse, Austin Chau, Antara Bhowmick, Shreenath Iyer, et al. 2018. Gunrock: Building a human-like social bot by leveraging large scale real user data. *Alexa Prize Proceedings*.

Hyojin Chin, Lebogang Wame Molefi, and Mun Yong Yi. 2020. Empathy is all you need: How a conversational agent should respond to verbal abuse. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–13.

Amanda Cercas Curry, Ioannis Papaioannou, Alessandro Suglia, Shubham Agarwal, Igor Shalyminov, Xinnuo Xu, Ondrej Dusek, Arash Eshghi, Ioannis Konstas, Verena Rieser, et al. 2018. Alana v2: Entertaining and informative open-domain social dialogue using ontologies and entity linking. *Alexa Prize Proceedings*.

Amanda Cercas Curry and Verena Rieser. 2018. #MeToo Alexa: How conversational systems respond to sexual harassment. In *Proceedings of the Second ACL Workshop on Ethics in Natural Language Processing*, pages 7–14.

Amanda Cercas Curry and Verena Rieser. 2019. A crowd-based evaluation of abuse response strategies in conversational agents. In *20th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, page 361.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Emily Dinan, Varvara Logacheva, Valentin Malykh, Alexander Miller, Kurt Shuster, Jack Urbanek, Douwe Kiela, Arthur Szlam, Iulian Serban, Ryan Lowe, Shrimai Prabhumoye, Alan W Black, Alexander Rudnicky, Jason Williams, Joelle Pineau, Mikhail Burtsev, and Jason Weston. 2019. The second conversational intelligence challenge (ConvAI2). ArXiv preprint arXiv:1902.00098.

Karthik Gopalakrishnan, Behnam Hedayatnia, Qinglang Chen, Anna Gottardi, Sanjeev Kwatra, Anu Venkatesh, Raefer Gabriel, Dilek Hakkani-Tür, and Amazon Alexa AI. 2019. Topical-chat: Towards knowledge-grounded open-domain conversations. In *INTERSPEECH*, pages 1891–1895.

Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear.

Eric J. Horvitz. 1999. Principles of mixed-initiative user interfaces. In *CHI '99: Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 159–166.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Dan Jurafsky, Liz Shriberg, and Debra Biasca. 1997. Switchboard SWBD-DAMSL shallow-discourse function annotation coders manual. In *Technical Report Draft 13, University of Colorado, Institute of Cognitive Science*.

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

Weixin Liang, James Zou, and Zhou Yu. 2020. Beyond user self-reported likert scale ratings: A comparison model for automatic dialog evaluation. ArXiv preprint arXiv:2005.10716.

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.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings of the Human Language Technology Conference of the NAACL*, pages 152–159.

Jan Pichi, Petr Marek, Jakub Konrád, Martin Matulık, and Jan Šedivy. 2018. Alquist 2.0: Alexa prize socialbot based on sub-dialogue models. *Proc. Alexa Prize*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI tech report*.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. *CoRR*, abs/1806.03822.

Hannah Rashkin, Eric Michael Smith, Margaret Li, and Y-Lan Boureau. 2019. Towards empathetic open-domain conversation models: A new benchmark and dataset. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5370–5381.

Chinnadhurai Sankar, Sandeep Subramanian, Christopher Pal, Sarath Chandar, and Yoshua Bengio. 2019. Do neural dialog systems use the conversation history effectively? an empirical study. *arXiv preprint arXiv:1906.01603*.

Skipper Seabold and Josef Perktold. 2010. statsmodels: Econometric and statistical modeling with python. In *9th Python in Science Conference*.

Abigail See, Stephen Roller, Douwe Kiela, and Jason Weston. 2019. What makes a good conversation? how controllable attributes affect human judgments. In *Proceedings of NAACL-HLT*, pages 1702–1723.

Andreas Stolcke, Klaus Ries, Noah Coccaro, Elizabeth Shriberg, Rebecca Bates, Daniel Jurafsky, Paul Taylor, Rachel Martin, Carol Van Ess-Dykema, and Marie Meteer. 2000. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational linguistics*, 26(3):339–373.

Anu Venkatesh, Chandra Khatri, Ashwin Ram, Fenfei Guo, Raefer Gabriel, Ashish Nagar, Rohit Prasad, Ming Cheng, Behnam Hedayatnia, Angeliki Metallinou, et al. 2018. On evaluating and comparing open domain dialog systems. *arXiv preprint arXiv:1801.03625*.

Joseph Weizenbaum et al. 1966. Eliza—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019a. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Thomas Wolf, Victor Sanh, Julien Chaumond, and Clement Delangue. 2019b. Transfertransfo: A transfer learning approach for neural network based conversational agents. *CoRR*, abs/1901.08149.

Dian Yu, Michelle Cohn, Yi Mang Yang, Chun-Yen Chen, Weiming Wen, Jiaping Zhang, Mingyang Zhou, Kevin Jesse, Austin Chau, Antara Bhowmick, Shreenath Iyer, Giritheja Sreenivasulu, Sam Davidson, Ashwin Bhandare, and Zhou Yu. 2019. Gunrock: A social bot for complex and engaging long conversations. ArXiv preprint arXiv:1910.03042.

Dian Yu and Zhou Yu. 2019. Midas: A dialog act annotation scheme for open domain human machine spoken conversations. ArXiv preprint arXiv:1908.10023.

Appendix

A Additional RGs

A.1 Coronavirus

Our bot has been available to users since early December 2019 – thus covering the entire period of the COVID-19 pandemic so far. As the pandemic progressed, users requested to talk about the pandemic more and more, and showed more interest in discussing home life rather than outside activites. We created the Coronavirus RG to meet users' expectations that our bot should be fully aware of the unfolding situation, on both an informational and emotional level.

The RG is activated when the user mentions or requests to talk about the Coronavirus. To be mindful of the stress that the pandemic has placed on many people, we never initiate the topic ourselves. We then give a comforting message acknowledging the emotional effect of the pandemic, and give the user the option to hear up to 4 news updates. After that, we ask an uplifting open-ended question, and handle the user's response by generating a response from our GPT-2 model (Section 5.2). We then perform a scripted transition to the Movies RG. For example after we've read out a news headline:

> **User**: *No thank you. I'm exhausted from listening to the news all the time.*
> **Bot**: *No problem. I know this situation has been very stressful for many people; I've read online that it can be helpful to remind ourselves of small, good things that have happened to us lately. What's something nice that's happened to you in April?*
> **User**: *Well it was my birthday!*
> **Bot**: *That's great. Did you have fun?*
> . . .

A.2 News

The News RG provides the user with news stories that interest them, and discusses them. We set up a cron job to query the Washington Post API[24] and scrape several news-based subreddits[25] every day, and place the data in an ElasticSearch index. When the user asks to talk about news, the News RG asks the user for a topic (e.g., *Joe Biden*), if it wasn't already supplied. The RG then queries the ElasticSearch index for news stories with the desired topic in the headline, selects the most recent one, reads out the headline to the user, and asks if they'd like to hear more. If accepted, we read out the first three sentences of the article.

Our original goal was to allow the user to ask follow-on questions about the article, and to answer them with a **Neural Question Answering** model. We hoped this would help realize our design goals of conversational phrasing and enabling user initiative (Section 1). To begin this process, the News RG would invite the user to ask questions. We then used the SpaCy coreference resolution module (Honnibal and Montani, 2017) to decontextualize the user's question with respect to the last two utterances from the News RG. For example, *how many votes did he win?* might be transformed to *how many votes did Joe Biden win?* The decontextualized question, along with the entire news article, was then sent to a BERT-Large model (Devlin et al., 2018) trained on the Stanford Question Answering 2.0 dataset (Rajpurkar et al., 2018) by HuggingFace.[26] The model would output either a span in the article, or 'no-answer' – meaning the question cannot be answered by the provided article.[27]

Unfortunately, in our internal testing, we found that this system had several substantial problems. First, errors in the coreference module were common, and would cascade to the QA module. Second, we found that users asked a very different distribution of questions, compared to the SQuAD training questions. For example, users were likely to ask more open-ended or causal questions (e.g., *what*

[24] An API call to scrape Washington Post news articles provided by Amazon Alexa.

[25] /r/News, /r/Sports, /r/Politics, /r/Futurology, /r/Science, /r/Technology, /r/WorldNews

[26] `https://github.com/huggingface/transformers`

[27] Since the article was often much larger than the maximum context size for BERT, we ran the model on chunks. Within each chunk, we discarded spans which were ranked lower than 'no-answer', then merged the answers and re-ranked by confidence of the predictions.

happened next?, *why did they do that?*). These are difficult for off-the-shelf QA models, which tend to excel in answering factoid-style questions. Third, users were likely to ask questions whose answers are not present in the news article. Though our model was trained on SQuAD 2.0 (which contains unanswerable questions), it would often choose an irrelevant answer that type-checks with the question, as Jia and Liang (2017) have also reported. Even when the QA model correctly classified unanswerable questions, we would have needed to build a substantial open-domain question answering system to handle these questions. Overall, these problems made our system a poor and unreliable user experience; requiring more time and effort to fix than we had available.

A.3 Other RGs

Launch Handles the first few turns of the conversation (introducing the bot and learning the user's name). An example can be seen in Table 1.

Acknowledgment When the user changes topic to a new entity, this RG uses the entity's membership in certain Wikidata categories to select a one-turn scripted acknowledgment (e.g. *Oh yeah, I read ENTITY last year - I couldn't put it down!* if the entity is a book). This RG aims to give a natural and conversational acknowledgment that a new topic has been raised, before handing over to another RG (e.g. Wiki/Opinion/News) to discuss the entity in more depth.

Alexa Commands Users often try to issue non-socialbot commands (such as playing music or adjusting smart home devices) to our socialbot. This RG detects such commands, informs the user that they're talking to a socialbot, and reminds them how they can exit.

Closing Confirmation Our bot stops the conversation when the user issues a command like *stop* or *exit*. However, users indicate a possible desire to exit through many other more ambiguous phrases (e.g., *do you just keep talking*, *what's happening*). This RG detects such cases using the *closing* dialogue act label (Section 4.2) and regex templates, asks the user if they'd like to exit, and stops the conversation if so.

Complaint Provides an appropriate response when a user complaint is detected. This RG uses the Dialogue Act classifier's *complaint* label to detect generic complaints, and regular expressions to detect misheard complaints (the user saying that Alexa misheard them), clarification complaints (the user saying that Alexa is not being clear), repetition complaints (the user saying that Alexa is repeating itself), and privacy complaints (the user saying that they don't want to share information). We wrote different responses for each type of complaint, to reflect understanding of the user's concerns.

Fallback Always provides a response (*Sorry, I'm not sure how to answer that*) or prompt (*So, what are you interested in?*) to be used when no other RG provides one.

One-Turn Scripted Responses Provides handwritten responses to common user utterances (e.g. *help*, *chat with me*, *hello*) that can be handled in a single turn.

Red Question Detects if the user asks our bot a 'red question' – i.e., a question we are not permitted to answer, such as medical, legal, or financial advice – and informs the user that we cannot answer. To recognize these questions, we trained a multinomial logistic regression model on bag-of-words features, using data from the /r/AskDoctor, /r/financial_advice, and /r/LegalAdvice subreddits.

B Tooling and Processes

B.1 Dashboard

We built a browser-based dashboard to provide ourselves with easy readable access to conversations and the associated metadata. The landing page shows aggregate rating statistics broken down by date and code version. The dashboard can filter conversations based on metadata such as number of turns, ratings, entities and RGs used. For each conversation, the dashboard displays important turn-level attributes, such as latency, entities, annotations, state information, RG results, and logs. It can provide a link pointing to a specific turn, which is very useful for discussions and issue tracking. The dashboard can rerun the conversation with the current version of our bot, to quickly test if our local changes fixed the problem. Aside from displaying conversations, the dashboard also has tabs to track errors and latencies, divided by severity level. Easy accessibility and visibility of errors made us more aware and likely to fix these errors quickly.

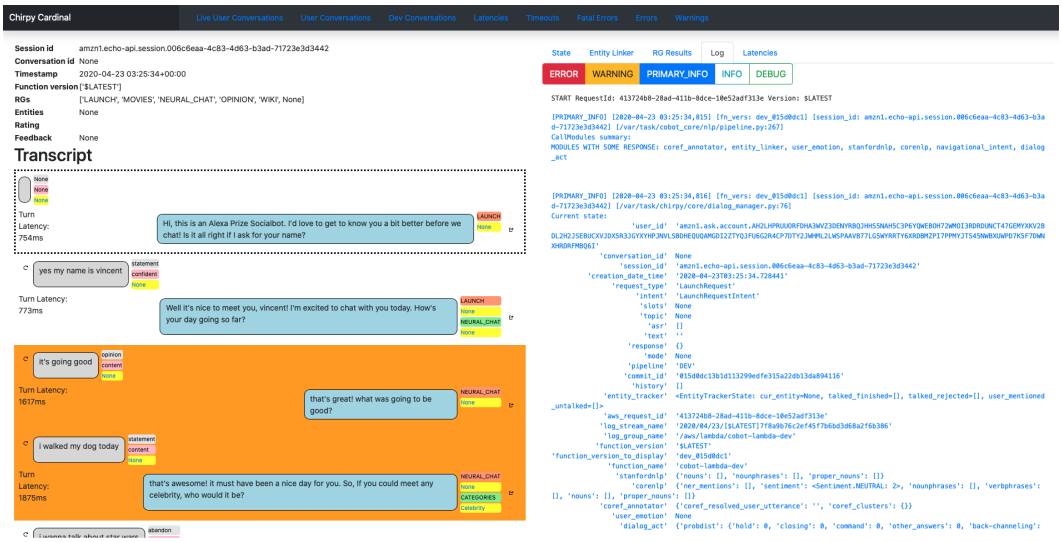


Figure 10: Screenshot of an example conversation (not with a real customer) in the dashboard. The tags next to each utterance are annotations from the bot. The background color of the utterance is the latency of that specific turn (white being normal and orange being slow). The pane on the right shows the logs for the turn.

B.2 Processes

Code Review We realized early on that maintaining high code quality is important for maintainability and extensibility. We set up a circular code review process to ensure that any code we write is understandable by another team member and adheres to certain quality standards.

Integration Tests We also instituted integration tests, to ensure that our bot maintains certain core functionality. We often found that some changes we made in one part of the bot had unexpected and damaging effects in another part of the bot; integration tests helped to catch these issues.

Canary Testing We had two versions of our bot – **mainline**, which handled real customers, and **dev**, which we used for developing new features. At first, new dev versions were solely tested by team members, before being pushed to mainline. However, especially as the complexity of the bot grew, this method became insufficient to identify problems in new dev versions – meaning that bugs were being discovered in mainline. We set up a canary testing framework, which directs a controllable percentage (typically 10%-50%) of customer traffic to dev. This was very useful in allowing us to tentatively test out new features with larger numbers of people, before deploying to all customers, thus protecting our ratings.

UX Officer Each week, we have a dedicated UX officer, whose primary responsibility is to monitor the conversations, identify problems, and get a sense of the strengths and weaknesses of the current system. This person is also responsible for alerting other team members to things that need to be fixed, and communciating their overall findings to the rest of the team at the weekly meeting. The role rotates every week so every team member has a chance to see the bot in action, and stay in touch with the overall user experience.

Sprint Planning and Issue Tracking We use Jira to track issues to be fixed – each is assigned to the person in charge of the relevant component. We have a weekly sprint planning meeting where we prioritize the most important things to work on over the next week, and use Jira to track the sprint.

C Dialogue Act Classifier

C.1 Modifications to Label Space

We modified this schema to better fit the needs of our bot, adopting 19 out of 23 dialogue act labels from MIDAS paper, and creating 5 new labels: *correction*, *clarification*, *uncertain*, *non-compliant*, and *personal question* to support UX-enhancement features such as the ability to respond to clarifiying questions. We dropped the labels *apology*, *apology-response*, *other*, and *thanks* since there were very few (n $\leq$ 80) examples of them in the original dataset and we rarely observed these dialogue acts in our bot.

C.2 Labeling Procedure

To create our gold-labeled dataset from our bot, we first determined which classes we most wanted to improve, based on per-class F1-Score for the baseline model and the new features we wanted to build. For example, since we wanted to improve our complaint handling, we prioritized this category. Next, we ran the baseline model on data from our bot to collect pseudo-labels. We randomly sampled 300 examples per label and then annotated whether the true label matched the predicted label. If not, we annotated what the correct label was. Using the pseudo-labels as a starting point increased efficiency, since the binary decision of "correct or incorrect" is much easier than the choice between 24 labels, and this method significantly reduced the number of non-binary decisions necessary. It also improved balance over classes, since it gave us greater control over the classes in the sample, and allowed us to prioritize certain categories. The result of training with gold-labeled examples is reported in Table 4.

D Emotion classifier and analysis

In order to understand and analyze users' emotions, we finetuned a RoBERTa model (Liu et al., 2019; Wolf et al., 2019a) on the EmpatheticDialogues dataset (Rashkin et al., 2019), which contains 24,850 examples broken into an 80-10-10 train-dev-test split. In particular, our training and test data consisted of the first utterance from each dialogue (as it is the only one with a label), along with its label (one of 32 fine-grained emotions, listed in Figure 11).

The RoBERTa model achieves a top-1 accuracy of 61.5% and an F1-score of 0.596. However, many of the misclassifications are due to the model choosing a label very similar to the gold label. For example, in the confusion matrix in Figure 11, we see that *angry* is often misclassified as *furious*, and *terrified* as *afraid*, among others. In contrast, the top-5 accuracy is 92%.

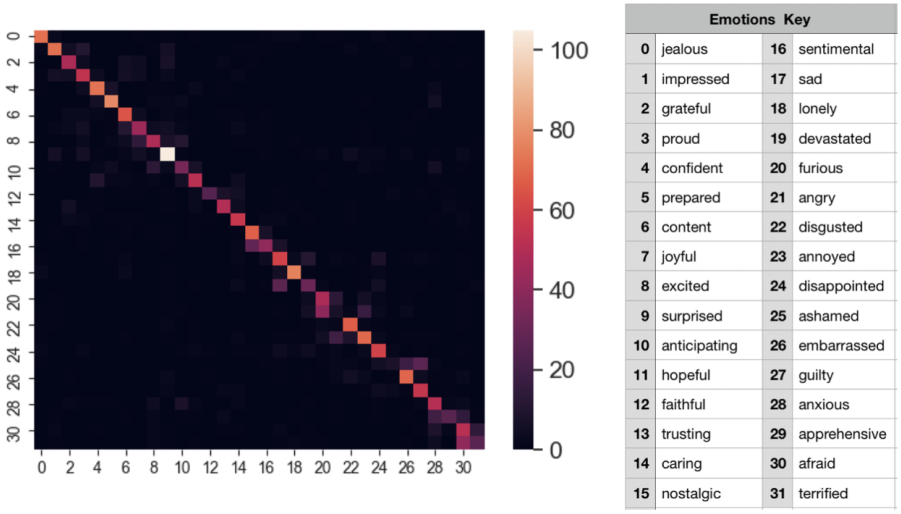
One difficulty in applying this classifier to our user utterances is domain shift. The EmpatheticDialogues training utterances all describe a strongly emotional personal situation in complete written sentences, in a self-contained way (i.e., with no preceding context) – for example, *A recent job interview that I had made me feel very anxious because I felt like I didn't come prepared.* By contrast our user utterances are spoken, typically not complete sentences, require conversational context to understand, and encompass many different dialogue functions (such as giving commands, answering questions, choosing topics, greeting and closing, etc.). Importantly, most utterances are emotionally neutral. As the classifier has no 'neutral' label, it assigns spurious emotions to these neutral utterances.

D.1 Relationship between Rating and User Emotion

To understand users' emotions and how they relate to our bot's performance, we replicated our experiment for dialogue act labels by applying a regression analysis, to the emotion classifier labels and the ultimate rating of each conversation.

Before performing this analysis, we removed all one-word utterances, since we assumed that these would not contain any emotion, and 66 common utterances that accounted for 40% of responses (e.g. *yes* and *no*), assuming that they were also neutral.

Figure 6 shows that, as we would expect, positive emotions have the largest positive coefficients and negative emotions have the largest negative ones. A possible explanation for the anomalies (e.g. "terrified" having a relatively large positive coefficient) is that the emotion classifier strongly associates certain entities with emotions and struggles to recognize when these entities are used in



Emotions Key			
0	jealous	16	sentimental
1	impressed	17	sad
2	grateful	18	lonely
3	proud	19	devastated
4	confident	20	furious
5	prepared	21	angry
6	content	22	disgusted
7	joyful	23	annoyed
8	excited	24	disappointed
9	surprised	25	ashamed
10	anticipating	26	embarrassed
11	hopeful	27	guilty
12	faithful	28	anxious
13	trusting	29	apprehensive
14	caring	30	afraid
15	nostalgic	31	terrified

Figure 11: Confusion matrix for RoBERTa emotion classifier.

different contexts. For example, it associates "tiger" with "terrified", even when "tiger" is in a positive context such as "I like tigers."

E Offensive User Experiment Details

E.1 Offense Type Detection

To determine the offense type, we hand-labeled 500 most common offensive utterances, which accounted for 53% of all the offensive utterances we collected to the date. We used 6 categories: sexual, insult, criticism, inappropriate topic, bodily harm and error. To classify the user utterance into one of these categories, we built regular expressions checking if the given user utterance contains one of the hand-labeled examples for an offense type. We then used the offense type to contextualize our `COUNTER+PROMPT` and `EMPATHETIC+PROMPT` responses.

E.2 Response Strategy Configurations

This section gives a detailed description of the configurations used in the Offensive User experiments (Section 5.9).

1. `WHY`: We ask the user why they made the offensive utterance (and this forms the entire bot utterance for the turn). The Offensive User RG responds with *OK* to whatever the user says next, then hands over to another RG to supply a prompt. For example: **Bot**: *Why did you say that?*, **User**: *because you weren't understanding me*, **Bot**: *OK. So, who's your favorite musician?*
2. `WHY+NAME`: Same as `WHY`, but we append the user's name to the end of the bot utterance. For example: *Why did you say that, Peter?*
3. `AVOIDANCE`: The bot politely avoids talking about the offensive topic, e.g. *I'd rather not talk about that.* This forms the entire utterance for the turn; the bot does not give any prompt to steer the conversation in a different direction.
4. `AVOIDANCE+NAME`: Same as `AVOIDANCE`, but we append the user's name to the bot utterance. For example: *I'd rather not talk about that Peter.*
5. `AVOIDANCE+PROMPT`: Same as `AVOIDANCE`, but we also give a prompt to change the topic. For example: *I'd rather not talk about that. So, who's your favorite musician?*

6. AVOIDANCE+NAME+PROMPT: Same as AVOIDANCE+NAME, but append a prompt to the end of the utterance. For example: *I'd rather not talk about that, Peter. So, who's your favorite musician?*
7. COUNTER+PROMPT: Strategy suggested by Brahnam (2005) and evaluated by Chin et al. (2020). In our interpretation of the strategy, we point out the inappropriate nature of the user utterance to the user, and attempt to move on to a different topic. For example, *That is a very suggestive thing to say. I don't think we should be talking about that. Let's move on. So, who's your favorite musician?*
8. EMPATHETIC+PROMPT: Strategy suggested by Chin et al. (2020) as the most appropriate strategy. In our interpretation of the strategy, we empathize with the user's desire to talk about inappropriate topics, and attempt to move on to a different topic. For example, *If I could talk about it I would, but I really couldn't. Sorry to disappoint. So, who's your favorite musician?*

For COUNTER+PROMPT and EMPATHETIC+PROMPT, unlike Brahnam (2005) and Chin et al. (2020), we handwrote responses tailored for the type of the user offense (e.g. sexual, curse, etc.)

Note that WHY and AVOIDANCE cannot be combined because they are contradictory, and WHY and CHANGE cannot be combined because we cannot ask two questions in one turn.

F Opinion Agreement Policy Details

Here we will show demonstrations of the 3 different Opinion Policies. All the demonstrations below are generated by researchers and are not actual user conversations.

F.1 ALWAYS_AGREE Policy

We begin by asking the user their opinion on the entity. Then, we agree with the user on their sentiment on the entity and share our reason, and ask the user their reason for liking/disliking the entity. After that, we propose another reason for liking/disliking the entity, and ask whether the user agrees with our reason or not. Finally, we switch to a related entity and repeat the same steps as described before. A demonstration of the policy is shown below.

User: *let's talk about cats*
Bot: *Ok! Do you like cats?*
User: *cats are my favorite*
Bot: *Sounds like you like cats. Me too! I feel like cats are so fluffy. What about you?*
User: *i like cats because they are very independent*
Bot: *That's so true. That reminds me of another reason I love cats. I feel like they treat you as an equal. Do you agree?*
User: *i totally agree*
Bot: *What about dogs? Do you like dogs?*
User: *yes*
Bot: *Me too! You know, I think the reason I'm a fan of dogs is because they love you back no matter what. What do you think?*
User: *yeah i feel the same way*
Bot: *Totally. I also like dogs because of how nifty they are. Do you feel the same way?*
User: *Yeah you are right they are pretty nifty.*
Bot: *Thanks for sharing! It's nice to know your likes and dislikes. Do you want to know more about dogs?*

F.2 LISTEN_FIRST_DISAGREE Policy

We initially ask the user for their opinion and reason (i.e. "What do you think about cats?" followed by "Why do you like cats?"), then we disagree with the user by giving our opinion and reason. After that, we switch to a related entity and agree with the user on their sentiment on the new entity. A demonstration of the policy is shown below.

User: *let's talk about cats*
Bot: *Ok! Do you like cats?*
User: *cats are my favorite*
Bot: *What's your favorite thing about cats?*
User: *i like cats because they are very independent*
Bot: *That make sense. I have to be honest though, I'm not a big fan of cats actually. I feel like they don't respect my personal space. Can we agree on that?*
User: *yeah i guess you are right on that one*
Bot: *What about dogs? Do you like dogs?*
User: *yes*
Bot: *Me too! You know, I think the reason I'm a fan of dogs is because they love you back no matter what. What do you think?*
User: *yeah i feel the same way*
Bot: *Totally. I also like dogs because of how nifty they are. Do you feel the same way?*
User: *Yeah you are right they are pretty nifty.*
Bot: *Thanks for sharing! It's nice to know your likes and dislikes. Do you want to know more about dogs?*

F.3 CONVINCED_AGREE Policy

This is a stronger version of the previous policy in which we immediately disagree with the user and give our opinion and reason (i.e. "What do you think about cats?" followed by "That's good to know, but I don't quite get the appeal of cats. I feel like they are too independent. But I want to hear why you like cats"). After receiving an answer from the user, we agree with the user and appear convinced. After that, we switch to a related entity and agree with the user on their sentiment on the new entity. A demonstration of the policy is shown below.

User: *let's talk about cats*
Bot: *Ok! Do you like cats?*
User: *cats are my favorite*
Bot: *Glad to meet a fan of cats! I have to be honest though, I'm not a big fan of cats actually. I feel like they don't respect my personal space. But I'm interested to hear why you like cats?*
User: *i don't agree i like that about cats because that's how they show their love*
Bot: *That make sense. Now that I think about it, there are a few things I like about cats. For example, they are very independent. What do you think?*
User: *yeah that's right.*
Bot: *What about dogs? Do you like dogs?*
User: *yes*
Bot: *Me too! You know, I think the reason I'm a fan of dogs is because they love you back no matter what. What do you think?*
User: *yeah i feel the same way*
Bot: *Totally. I also like dogs because of how nifty they are. Do you feel the same way?*
User: *Yeah you are right they are pretty nifty.*
Bot: *Thanks for sharing! It's nice to know your likes and dislikes. Do you want to know more about dogs?*