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 activities. 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 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. 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.

---

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.
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 communicating 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 clarifying questions. We dropped the labels apology, apology-response, other, and thanks since there were very few (n ≤ 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 \cite{Liu2019,Wolf2019} on the EmpatheticDialogues dataset \cite{Rashkin2019}, 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
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)](Brahnam2005) and evaluated by [Chin et al. (2020)](Chin2020). 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)](Chin2020) 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?