ZotBot: Using Reading Comprehension and Commonsense Reasoning in Conversational Agents

William Schallock* **Daniel Agress** **Yao Du** **Dheeru Dua**
Lyuyang Hu **Yoshitomo Matsubara** **Sameer Singh**
University of California, Irvine CA
{wschallo,sameer}@uci.edu

Abstract

We describe the ZotBot system for open-ended conversations, designed for the Alexa Prize competition. We focus on two main shortcomings in existing conversational agents: lack of awareness in commonsense reasoning when responding to user utterances (resulting in nonsensical or uninteresting responses) and inability to understand semantics and converse naturally about fact-based articles in a compelling manner. First, we combine existing work in commonsense KBs, pretrained language models, and graph completion models to generate natural and intuitive responses, consistent with our commonsense knowledge, for open-domain utterances. Second, we utilize question generation models for both reading comprehension and conversational followups for discussing fact-based articles. These contributions are implemented within a system engineered to be modular, allowing easy injection of manually-scripted responses, as well as supporting a detailed logging and analysis system. We present examples and analyses highlighting the benefits and shortcomings of ZotBot, and conclude with the lessons learned for future research.

1 Introduction

In recent years, developing automated systems that are able to hold open-ended conversations with humans has become increasingly relevant. Understanding responses, reading relevant sources, performing commonsense reasoning, and consequently maintaining coherence, user interest, engagement, topicality, and empathy in generating responses is very challenging even for the state-of-the-art artificial intelligence and natural language processing systems. Furthermore, advancements in such open-ended conversation agents can have a significant impact for downstream applications in commercial virtual assistants (like Alexa and Siri), therapy, education, and even task-oriented chatbots. For these reasons, there have been a number of shared tasks and workshops in the scientific community [Rastogi et al., 2019, Chen et al., 2019], the most prominent of which is the Amazon Alexa Prize [Khatri et al., 2018c].

Among the many challenges that are faced by current conversational agents, being able to generate natural responses and continue the conversation in interesting ways are two of the most important ones. Generating such responses is difficult because it requires both a semantic understanding of the topic and the context (for *any* domain) and to generate a non-obvious yet coherent/relevant response (for *any* prompt). Existing approaches solve this task by either targeting a narrow set of popular topics, such as movies [Chen et al., 2018] and politics [Yoshino and Kawahara, 2015], or by generating specific responses that *lead* the users down specific dialog flows. Both of these result in conversations that are unnatural, stilted, and often lead to a poor user experience [Dambanemuya and Diakopoulos].

*Apart from the first and last authors, the authors are listed in alphabetical order.

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

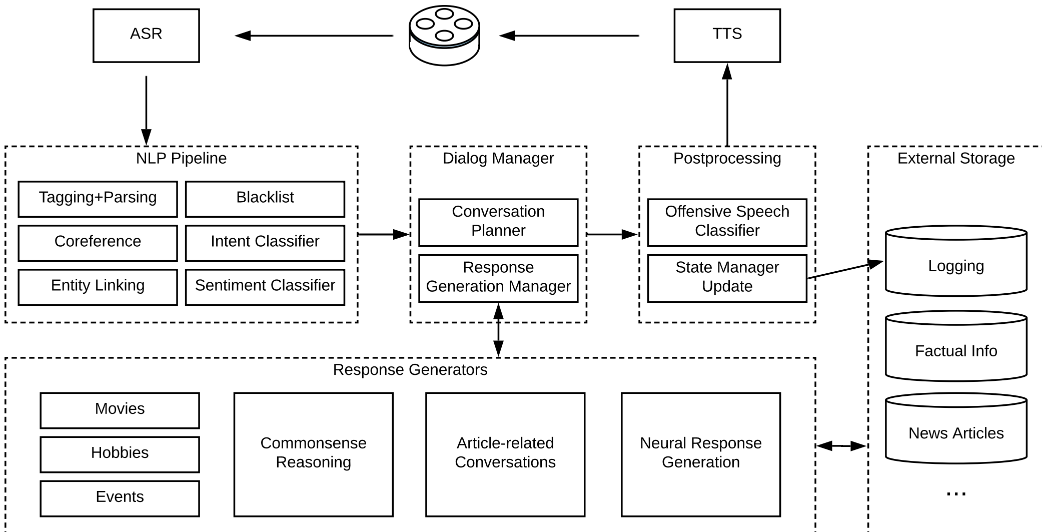


Figure 1: **ZOTBOT System Architecture:** consisting of the NLP pipeline (Section 2.1) and the blacklist module (Section 2.2), along with a number of *response generators* (RGs) such as the manually-scripted ones (Section 3), common-sense reasoning (Section 4), reading comprehension of articles (Section 5), and the backup neural response generator (Section 6).

In this paper, we describe our ZOTBOT system, our entry into the 3rd Amazon Alexa Prize [Khatri et al., 2018a]. Our contributions focused on building upon, and adapting, sophisticated natural language processing models, along with selective manual intervention and annotations, to generate natural responses. Part of our agent focuses on performing *commonsense reasoning* that provides natural and intuitive responses to any "general" topics (such as talking about activities users like to do), by combining recently introduced knowledge bases [Sap et al., 2019], graph completion models [Bosselut et al., 2019], crowdsourcing, and neural language generation models [Radford et al., 2018]. For more factual prompts (such as discussions around news articles), we combine reading comprehension [Norvig, 1986] models, large datasets of questions and answers [Kollar et al., 2018], and neural language generation [Gao et al., 2019] to generate accurate answers to queries and interesting followups. We put these different open-domain response generators into a system that also facilitates easy addition and customization of closed-domain response generators, to provide combine the high coverage of general systems with high-precision, tailored responses for a better user experience.

To evaluate our contributions, we implemented these ideas in the Amazon Cobot framework [Khatri et al., 2018b], as part of the Alexa Prize competition (we describe the overall system architecture in the next section). In summary, feedback from the users indicated they found the system to be often quite impressive when it came to individual responses to their prompts (for both manual and automated techniques). However, our bot struggled considerably with transition between the response generators. Due to these challenges (and other lessons learned that we describe in Section 8), ZOTBOT did not perform well in the competition.

2 System Design and Architecture

In this section, we describe the overall architecture of ZOTBOT. Figure 1 illustrates the system architecture, showing the main components of ZOTBOT, i.e. the input processing, dialog manager, response generators, post-processing, and external storage for these components. We will describe the pre- and post-processing components, and discuss the response generators in the later sections.

User utterances are provided via Amazon's ASR service, which converts the user's spoken dialog to text. This text is then processed by the NLP Pipeline, which we will describe in Section 2.1. The NLP components output their information to the dialog manager's state manager. The dialog manager has two primary tasks, selecting how to respond and formulating the response, which are respectively carried out by the conversation planner and the response generation manager. The conversation planner utilizes the features in the state manager, such as the output from the NLP Pipeline and the previous turn's state, to select all the possible candidate response generators. Once the dialog manager has selected the candidate response generators, all the candidate response generators are run in parallel by the response generation manager and return a response. If multiple candidate response generators return a valid response, the response generator manager is the final arbitrator that selects which of the candidate response to select as the bot's response.

ZOTBOT's responses are formed by response generators. There are two types of response generators, domain-specific response generators, and general response generators. Domain-specific response generators are specialized to generate responses in a specific topic domain while general response generators are designed to carry out an additional tasks (such as greeting the user) or to generate responses that do not fall into a topic our bot can handle. Each domain-specific response generator has a callback to multiple response handlers that are used to craft different pieces of the specific response type, and often use external storage lookups (such as reading an article or retrieving facts about a movie). Response generators are designed for multiple turns, and they can provide responses that are lower priority, or generate higher priority ones that override other response generators. General response generators also use a similar response handler design, however, the handler response type and selection vary based on the purpose of the response generator in question.

Once the handlers have generated the response, the response generator performs a set of clean up tasks, such as resetting handlers and sorting the various article threads that have been visited. The response generator manager selects the final response and updates the state manager and DynamoDB. Outside the of the ZOTBOT's main architecture, there are also external processes that are running that continually update the knowledge sources in external storage.

2.1 Natural Language Processing Pipeline

Here, we describe the core techniques used in our NLP pipeline.

Noun Phrase Extraction and Part of Speech Tagging The main objective of our NLP pipeline is to process or analyze users' utterances as part of the upstream process, and provide downstream modules with a set of high-level information about the utterances. For instance, noun phrases extracted from the utterances will be helpful for searching specific entities such as professional sports players, actors, and actresses. Similarly, POS (Part of Speech) tags inferred from utterances (*e.g.*, POS tags of "how are you" will be `adv`, `verb`, and `pron` respectively) may help us analyze relations between entities in the context. For these tasks, we use an off-the-shelf package spaCy, and its small model trained on English corpus ("en_core_web_sm") [2] due to its fast and accurate models, with an easy-to-use API.

EVI Amazon's EVI service[3] is a knowledge base and semantic search engine software that specializes in answering common queries. EVI works by providing a text query, such as "How tall is Everest?", searching the EVI knowledge base, and finally returning the answer as text, i.e. "Mount Everest is 29,029 feet above sea level". EVI was called via an API service in the NLP Pipeline, with the user's utterance as the query, and the output would be placed in the State Manager. Both the dialog manager and the response generators could utilize this output in the bot's response.

AIQL Alexa Information Query Language (or AIQL) is an API query service that allows ZOTBOT to access a subset of the Alexa Information Knowledge Graph. While EVI is limited to a single return value, AIQL can be used for custom responses for each named entity in the knowledge graph. We use AIQL to obtain a list of triples that relate a named entity to a property, given a specific relationship called an ontology. The query API also supports entity resolution, and the ability to find related entities, i.e. entities that share properties. AIQL was utilized by some of our topic-based response generators, such as our film response generator, as we will describe later.

Coreference Resolution As we talk to socialbots, many of the user utterances might be *stand-alone*, where little context is needed to respond appropriately, such as "who is the current us president".

[2] https://spacy.io/models/en
[3] https://www.evi.com/

However, as the conversation proceeds, the user utterances often contain followups that rely on previous dialog for a complete understanding, such as “how old is he” as a followup to the previous utterance. Such follow-up questions, hotrumpwever, are difficult to answer as the word “he” in the example needs to be resolved, i.e. to “the current us president”. We address this problem by rephrasing user utterances before sending it downstream using coreference resolution [Ng, 2010], and replacing mentions with their antecedents, if any, i.e. replacing “he” in “how old is he” with “the current us president” (or “donald trump”, if our previous response identified him as the current president). Specifically, we use the *neuralcoref* package[4], that is an implementation of Clark and Manning [2016]. One of the primary concerns with coreference resolution is that it gets much slower and loses accuracy for larger contexts (as we hope our dialogues will be) and more mentions to resolve. To make coreference resolution efficient, we follow a “simple then complex” approach, in particular, we resolve only one entity from the user utterance and start with one previous utterance (user’s or ZOTBOT’s), and incrementally adding more till we discover an antecedent (up to a maximum of three previous context utterances).

Entity Linking With entity linking, we want to identify real-world entities that the user has mentioned in their prompts, primarily for suggesting related articles to continue the discussion with users. It includes an ElasticSearch engine containing two indices: name to alias mapping (NE) from Freebase knowledge base and news articles tagged with entity names (ART). Each user and ZOTBOT utterance is queried against NE index to identify the entities which are added to the state table. The history of entities are further used to suggest a new relevant article whenever its appropriate to respond and a relevant article with high confidence is found in ART.

2.2 Blacklist Filtering via Sentence Embeddings

To flag inappropriate user utterances (and detect potentially offensive content in our own ZOTBOT responses), we implement a “blacklist filtering” component in the NLP pipeline, partly using a profanity checker provided by Amazon [Khatri et al., 2018c]. When a user utterance is flagged as being inappropriate, ZOTBOT can respond appropriately or change the topic.

One of the problems with creating a blacklist is that while it is easy to match against a list of inappropriate words (a lexicon), there are many syntactic variations which are hard to define upfront, for instance, "Should I invest in Amazon stocks?" and "I wonder if I should buy shares of Amazon?" making simple regular expression matches brittle. To this end, we index some handcrafted utterances, their Universal Sentence Embedding [Cer et al., 2018], and corresponding responses to an Elastic Search index. Each user utterance is first matched with text utterances to get top-k matches (based on a confidence threshold) and then a nearest-neighbor search (based on cosine distance) is performed on these filtered matches. We employed a two-phase search strategy to reduce computation time while performing nearest neighbor match. Through a series of internal tests, our team verified that this approach did substantially reduce the number of false-negative cases for flagged inappropriate content. However, it required a few iterations of curating the list of prototypical sentences and playing with the threshold before we were satisfied with our false positive rate.

Use in Response Generators The semantic matching capabilities of this module also proved useful for handling several other functionalities effectively and efficiently, without much engineering overhead. First, in the commonsense reasoning module (described in Section 4), this module was utilized to search for human-written utterances for various activities such as video games and baseball. Second, in the question-answering module (described in Section 5), this module was used to find the generated question closest to the user utterance. We elaborate on these in their respective sections.

2.3 Debugging and Iterating Pipeline

The large scope and fast-paced nature of this project meant that our team members would need to efficiently develop and concurrently test each others’ code. To do this, our team utilized a development pipeline to write and review new code, AB testing to test the user engagement of new features, and a logging system to address software bugs.

[4] `https://github.com/huggingface/neuralcoref`

Development Pipeline: To allow iterative design, our team utilized two concurrent pipelines: production and beta. Throughout the competition, the importance of having a robust code review and internal testing system became clear, and our team adapted our iterative process to incorporate lessons we had learned, such as a more structured code review (this is discussed in more detail in Section 8).

AB Testing: In addition to the production and beta branches, our team made use of cobot's AB testing capabilities to test out the addition of new features. With the addition of multiple versions of our bot, it became a bit more difficult to track down and replicate software bugs. To make it easier for our team to debug issues present in a specific version of our bot, our team modified cobot's code which selects the AB version at run time, so that it would be able to select a version from the command line interface.

Logging and Analytics: As with any kind of software, having an expedited means of documenting and fixing bugs is crucial. To make the debugging process simpler, we implemented a system to log all errors encountered during run time in the state manager. While cobot does offer debugging logs, our team realized that the logs are often too broad to be able to track down the root cause. We refactored our code to add try-except blocks to every individual process in ZOTBOT so that logging would contain more details (such as values of variables). Outside of the main ZOTBOT pipeline, we ran a cron job to collect all the logging information and exported it to a CSV file, which can be easily consumed by standard plotting and visualization libraries. These files thus expedited the debugging process as we could visualize the distribution, breakdown, and duration of all the response generators.

3 Manually-Scripted Response Generators

Although the primary objective of ZOTBOT was to be able to converse in an open-ended manner, many users often want to chat about a select set of topics. For our bot to converse naturally about popular topics such as movies, activities, and one-off current events, we developed a few "manually-scripted" response generators. These response generators are template-based response generators that could converse about common entities in these domains. In this section, we will detail the approach and some of the shortcomings we encountered with these response generators.

To facilitate the development of these *specific* response generators, we designed the ZOTBOT pipeline to be modular, such that these RGs can be introduced, modified, or removed easily, with little disruption to the existing pipeline. Further, we developed tools to query relevant KBs, set up crowdsourcing tasks, or easy-to-use interfaces for our team members to edit and inject the scripts. Three response generators that benefit from the manual scripting are movies, activities, and one-off events.

Movies: To generate responses about movies, we utilized AIQL to retrieve knowledge graph triples pertaining to a specific movie or actor/actress. The triples were used to populate the slots in the movie response generator's templates, and using AIQL proved useful for formulating a response about the attributes of a movie, such as its genre or related franchises. The attributes could also link to other entities such as additional movies in an ontology.

One-off events: For one-off events, a limited number of handwritten templates were created, and the bot would craft its response by cycling through the templates. Each one-off event supported the addition of an added level of response parsing that could extract event relevant information from the user's response. One example of this is for the Superbowl, where the entity-linker added support for ASR token mismatching for team names (i.e. "forty niners" and "fort eh niners" to "49ers"). If the one-off event implemented a specific event response parser, the output of the parser could be used to create a branching dialog, in which specific templates could be selected based on the user response.

Other RGs: We also include a number of other "special case" response generators, mostly single-turn, that we will describe only briefly. We have a "No Answer" response generator that fires when the model does not know what to do (we override it with the NRG, described in Section 6). We built a "Alexa Skill" response generator to let the user know we cannot handle non-socialbot Alexa commands (e.g. "order more socks"). Last, we create a few very specific response generators, together called "Misc.", to carry out specific tasks such as switching topics and helping when the user is confused.

Examples We present a few examples of such manually-scripted response generators.

Movies: Below is an example[5] of the movies response generator.

ZotBot: what would you like to talk about?
User: *movies*
ZotBot: Sure. Have you seen anything interesting recently?
User: *rise of skywalker*
ZotBot: That's Great. How did you enjoy watching it?
User: *i would give it an solid 9 out of 10*
ZotBot: The plot was a bit chaotic. I feel like the pacing could have been improved if they simplified things. It felt like there were too many plot threads to follow at once. What did you think about the plot?
User: *the plot was fast but manageable*
ZotBot: What was your opinion of the big kiss? I have heard mixed reviews from others I have chatted with.
User: *i liked it i think they have good chemistry what do you think*
ZotBot: I wasn't a fan of the kiss. The whole romance between them seemed a bit forced. It was interesting to see what they did with the force bond, but it could have been used for more philosophical means, like duality.
User: *fair enough*
ZotBot: Did you enjoy the film's music?
User: *yeah the whole sound track was really well done*
ZotBot: I thought the music was excellent. I loved the way that the composer John Williams mixed in the new music with the iconic themes from past movies.
User: *yeah*
ZotBot: Discussing sci-fi works reminds me of Jurassic Park. Have you seen it?

This example showcases specific content we had written about Star Wars Episode IX that included a set of opinion question and answer response pairs that ZotBot could select from (this movie came out during the beta phase of the competition). The specific movie content was added to enhance the general movie dialog our bot was capable of having. Overall we noticed that users responded well to our addition of specific movie opinions, and tended to have longer responses.

Popular Activities: Below is an example conversation about video games, a popular topic.

User: *let's talk about video games*
ZotBot: I always feel very relaxed and refreshed. when I'm playing. How do you feel when you're playing video games?
User: *it makes me feel calm*
ZotBot: What sort of games do you like to play? I tend to prefer playing games that are highly technical and competitive.
User: *i like to play strategy games like civilization*
ZotBot: I always lose track of time whenever I play video games.
User: *yeah it can be tough to take a step a back*
ZotBot: Do you ever feel tired when playing? It's important that you take a break after playing for a long time to refresh your mind.
User: *i try to*
ZotBot: Do you always want to win the game? Personally, I just want to be social and have fun.
User: *i like to have fun*

This is a prime example of the kinds of conversations ZotBot could have about popular activities, as it showcases an array of bot responses that were crowdsourced from the MTurk hit. The dialog is typically focused on asking the user questions about the activity. While users generally appeared to be engaged by this type of conversation, the rigid dialog structure limited the user's ability to ask our bot questions or redirect the conversation.

Super Bowl: Below are two examples ZotBot conversation about the Super Bowl, which was implemented as a one-off event. The first is a pre-game conversation, and ZotBot's dialog focuses on projections and opinions, and the second example took place during the game (half-time) and focuses more on talking about recent plays and predictions for the rest of the game. Overall we noticed that users were particularly engaged with ZotBot right before and during the game; ZotBot averaged a 4.5 rating during that time.

[5] This, along with every example in the paper, are not the actual user conversations, but are examples provided by us that are similar to actual user interactions.

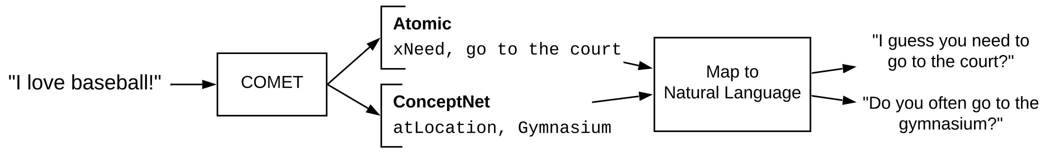


Figure 2: The two-step process involved in incorporating commonsense knowledge - generating the knowledge from a knowledge base and then translating it into natural language.

ZotBot:	What would you like to talk about?
User:	*the super bowl*
ZotBot:	Are you rooting for the 49ers, or the chiefs?
User:	*chiefs what about you*
ZotBot:	Well, I spend a lot of my time in Seattle, so naturally I'm a Seahawks fan. Considering the 49ers are one of our main rivals, I am rooting for the chiefs to win. Personally, I think the the 49ers are mostly going to run the ball, while the chiefs will focus on passing. What do you think will happen in the game?
User:	*so how do you think the game will go*
ZotBot:	I think it will be a close game, and it will come down to which team will have a better defensive line. I predict the chiefs will eek it out in the second half.
User:	*i am impressed how did you know that*

ZotBot:	What would you like to talk about?
User:	*the super bowl fifty four*
ZotBot:	It was a really close game going into half time. Are you rooting for the 49ers or the chiefs to pull ahead?
User:	*i am rooting for the chiefs*
ZotBot:	Both teams had a solid defense in the first half. What did you think of the game so far?
User:	*it is a good game so far*
ZotBot:	I was impressed by San Francisco's drive that ended in a run made by number 44. I think the 49ers will try to keep running the ball.
User:	*yeah i think that will be a safe bet*

Shortcomings While creating manually-scripted response generators proved to a useful means of supporting conversation about common topics, implementing such response generators does come with some notable drawbacks. First, manually-scripted response generators that relied on handwritten responses, like one-off events, require a significant effort to author and maintain the content, making it difficult to quickly add and update content. This is especially problematic for sudden events, like the crash of Kobe Bryant's helicopter, and events that were rapidly changing in an unpredictable manner, such as the initial stages of the coronavirus outbreak. Second, manually scripted response generators are too rigid and could not always adapt to the user requests. While our response generators were not purely linear, and we implemented a number of branching dialog structures, the users often still responded in unanticipated ways. Last, since the content of these manually-scripted response generators had to be manually authored, they often became stale and obsolete. We prioritized popular events, however either relied on user transcripts to identify these popular events, at which point they might not be popular anymore, or we had to anticipate them, which is impossible for all but the most obvious or scheduled ones (such as the Super Bowl).

4 Commonsense Reasoning for Open-Domain, Sensible Responses

Commonsense knowledge – facts about the everyday world - are an essential component of conversation [Young et al., 2018]. In conversation, people use their knowledge of the properties and relationships between entities to form opinions and ask questions about their own and their conversational partners experiences. For example, when one says that a funeral is a sad event or that after finals, one must be exhausted, one is relying on their commonsense knowledge. When conversing with a conversational agent, they expect some level of commonsense responses, and would find a joke in the former case extremely inappropriate.

We wished to incorporate commonsense knowledge into our ZotBot's responses, in an open-domain manner. In particular, our goal was that if the user brought up a commonsense topic or experience,

Table 1: Several relations from the ConceptNet and Atomic knowledge bases. The Atomic subjects are generally of the form PersonX __. The ConceptNet subjects are objects or activities.

Knowlege Base	Relations	Meaning
Atomic	`xAttr`	PersonX is seen as
	`xIntent`	Because PersonX wanted
	`oWant`	As a result, others want
ConceptNet	`AtLocation`	X is the typical location of Y
	`HasSubevent`	Y is a subevent of X

typically an activity they like to perform (like hobby) or occupation, ZotBot would be able to give a relevant, grammatical, and detailed response that demonstrates ZotBot's commonsense knowledge of the topic. We feel that this would enrich ZotBot's conversation by allowing it to engage in conversations about various activities, such as cooking or skiing, and personal events, such as going to weddings or graduations. For example, if the user utterance was "I enjoy cooking on my free time," we would hope to be able to ask questions about which ingredients they use or which cuisine they like. Such questions require the bot to incorporate commonsense knowledge. While such responses could be programmed manually for several domains, it was unfeasible to do this for the wide range of interests which users brought up.

Incorporating commonsense knowledge into our bot involved two main steps, as shown in Figure 2. First, we need to find a relevant and interesting piece of commonsense knowledge to inject into the conversation, for which we turn to existing commonsense knowledge bases[6]. Second, we need to translate this identified knowledge into a grammatical and engaging response for ZotBot to say, for which we use two different strategies, crowdsourcing and manually-generated templates.

Commonsense Knowledge Bases Several large knowledge bases exist which have gathered large quantities of commonsense knowledge. Two prominent examples are ConceptNet [Speer et al., 2017] and Atomic [Sap et al., 2019]. Both of these are collection of subject-relation-object (SRO) triples. The relations such as "is seen as" or "is used for" are *general properties* of a subject and objects define the property values of the subject. ConceptNet has 10^8 triples, and focuses on taxonomy of entities, "parts" of entities, and locations of entities and activities, e.g. "Basketball is played on a basketball court." Atomic has 10^6 triples and focuses on inferential reasoning and "states" of entities (primarily people), e.g. "If PersonX hurt PersonY, then PersonX is seen as upset." We refer to Table 1 for a list of some of the relevant commonsense relations contained in each of these two databases. Since these manually curated knowledge bases are often incomplete, they suffer from severe sparsity (many missing facts) as well as require explicit alignment of text to the knowledge bases. As a solution to both, we use COMeT [Bosselut et al., 2019], a neural network that is trained on ConceptNet and Atomic to generate new edges that are missing from these knowledge bases, as well as support textual input, by being built upon pretrained language models like GPT [Radford et al., 2018]. Using COMeT, we are able to use an arbitrary string as the subject, pick a relation from the KB, and receive an object as output. For example, if a user said "I got a hundred on my test", we input "PersonX got a hundred on their test" and receive "PersonX is seen as smart" from COMeT.

For a given user utterance, we need to find a relevant triple from out database to respond with. We used two primary strategies. First, we indexed knowledge base entries for several hundred common activities, such as video games, cooking, drawing etc. For all these activities, we pulled existing triples from ConceptNet and Atomic, or generated new edges using COMeT. We then did a keyword search (using sentence matching from Section 2.2) in user utterances to see if they had brought up any of these common activities. While this is effective for the covered topics, we found that it did not give broad enough coverage for the range of topics that users brought up. In addition, these triples were only matches for a keyword that the user brought up, and did not match the user's specific utterance. Thus, in this strategy, "I play baseball," "I watch baseball," or "I hate baseball" would all receive the same response. To address these issues, we augmented our indexed topics with on the fly generation of knowledge base triples using COMeT, which used the actual user utterance as subject.

[6]Although such results might also be obtained with a fully neural response generator, the advantage of using a knowledge base is that one has a better control and understanding of what is being said.

Table 2: Performance of commonsense response generators, compared to a manually-crafted, pre-written response generator. COMeT responses are generated on the fly using the COMeT knowledge graph and templates.

Response Generator	Avg Rating	Avg Duration	Avg Turns
All Conversations	3.08	253s	-
Manually-scripted	3.16	222s	4.26
Template-based	3.07	328s	1.81

Generating Natural Language Responses For the common activities we had indexed, we set up a Amazon mechanical-turk task which asked crowd-workers to choose a set of triples they think are relevant while chatting with a friend and write a diverse set of utterances based on the chosen triples. We will release these annotations at `https://uci-zotbot.github.io/` for research.

We employed two main strategies to respond to users while discussing a specific activity: manually-scripted response generator, template based response generator.

- **Manually-scripted response generator** first identifies the activity being discussed and then looks-up utterances created by the crowdworkers at random from the index to be used as responses.
- **Template-based response generator** used pre-defined templates which were populated based on COMeT's predicted edges for a new activity, not part of the indexed set. Although the quality was lower than the crowdsourced responses, these templated responses were able to reply to a wider range of user utterances.

Examples We present some examples of the two response generators. We first give a walk-through of how the response generators would respond to an utterance. We then give a sample conversation for each of the response generators.

First, let us look at an example where the user utterance was "I love playing baseball." Our *manually-scripted response generator* would find the word "baseball" in the utterance and would know that the user wanted to discuss baseball. It would then look up the crowdsourced sentences relating to baseball, and respond with one of them to the user. This response might be "Playing baseball is a great way to spend time with friends. It is also fun to win. Do you own a bat and glove?" As a fallback measure, we would run the phrase "PersonX loves playing baseball" through the *template-based response generator*. COMeT might generate the SRO triples:

1. `PersonX loves baseball, xNeeds, PersonX goes to the park`
2. `PersonX loves baseball, xAttr, PersonX is athletic.`

We could then respond, "Wow. You must be athletic. Are you planning on going to the park?" in case we the manually-scripted response generator does not yield anything.

Here is an example conversation that employs both the response generators together

ZOTBOT:	What do you like to do for fun?
User:	*I love traveling.*
ZOTBOT:	Travelling is great for the curious (is seen as curious) and adventurous (is seen as adventurous) who want to explore the world around them. How long are you home (wants to go home) until you become eager to start a journey?
User:	*I will leave for England in several weeks.*
ZOTBOT:	What do you like to explore (wanted to explore) when you travel? The best part of traveling is being able to see the world (wanted to see different places) and meet new people.
User:	*I enjoy learning about the history of the places that I visit.*
ZOTBOT:	Have you ever lost your luggage while traveling? (needed to pack their bags)
User:	*One time the airline lost my luggage, but they found it a few days later.*
ZOTBOT:	Wow. that sounds careless (is seen as careless). Did you have to go to the airport? (needed to go to the airport)

Experiments We present evaluation of the effectiveness of the common sense response generators.

Comparing to other response generators: We measure how the common sense response generators performed relative to other response generators. We see that the manually-scripted response generator performed well in both ratings and number of turns while the template-based performed poorly. As

Table 3: Comparing the response quality of crowdsourced responses to the COMeT generated responses. We measured the quality of a selection of crowdsourced and templated responses along the three metrics of grammar, relevance, and how interesting they were, rated are on a 1-5 scale.

Method	Grammar	Relevance	Interestingness
Manually-scripted	4.9	4.1	4.4
Template-based	3.9	2.6	2.8

we will discuss below, we attribute this to a high rate of false positives and offensive responses, which caused negative ratings and user disengagement.

Comparing manually-scripted and template-based RGs: For this experiment, we internally ranked the responses to 20 user utterance/bot response pairs from manually-scripted sentences and template-based utterances. We ranked the responses on three metrics: grammar, relevance, and interestingness. We concluded that the manually-scripted sentences were much better in all three metrics, and that our best strategy was to increase the coverage of the manually-scripted sentences. However, the template-based RG provided a greater coverage in many cases where it was not possible to have manually written sentences. In future it would be best to create a neural response generator trained on the crowdsourced data.

Shortcomings When making each of our two commonsense response generators we faced a number of challenges and shortcomings. We discuss a number of them below.

Lack of Coverage: Our biggest challenge was finding a generic set of activities we could connect with the user on. Often the user would bring up topics for which we had no indexed coverage and which COMeT had a difficult time interpreting. At these times, our bot was unable to respond effectively. We responded by trying to expand coverage and by creating backup responses to deflect the conversation when we did not recognize the topic the user was trying to discuss.

False Positives: At other times, our bot misinterpreted user utterances as relating to commonsense knowledge when in reality they were unrelated. This results in a confusing experience for the user, as the bot's response appears to be unrelated to the initial user utterance. For example, at one point, when users would mention "talking," the template-based response generator thought that they were referring to talk shows and responded accordingly. We address this to some extent by requiring stronger keyword matches for the indexed responses.

Grammar Mistakes: Due to its templated language generation strategy, COMeT's grammar was often incorrect. For example, a COMeT response discussing football was "Do you expect to fumbling the ball?" Additionally, the repetitive nature of the templates often made the conversations seem unnatural.

Content Mistakes: Being trained on just language modeling objective, COMeT often had difficulty performing correct activity linking, resulting in incorrect responses. For example, COMeT interpreted "playing bridge" as relating to construction, rather than as a card game.

Offensive Content: Occasionally COMeT would insult users ("You must be confused."), leading to low ratings. We ran our blacklist module (Section 2.2) on responses to make sure only appropriate responses go through.

5 Article-related Conversations Using Reading Comprehension Models

Apart from specific entities and general topics that are handled by the response generators described so far, there is a large fraction of prompts and queries where the users want to converse about factual information. For example, they may want to discuss a recent news story or talk in detail about a specific prominent entity (whose information is available on Wikipedia). Two allow for natural and useful conversations for such topics, we identified two primary challenges for ZOTBOT. First, since reading out the whole article does not make for an interesting experience, we want to be able to answer any specific queries that the user may have about the article, under strict computational constraints. Second, we want to be able to converse about the article even if the user does not have specific queries for the system, and do so in a natural, engaging and fluid manner. We build upon recent work in *question generation* to address both these goals.

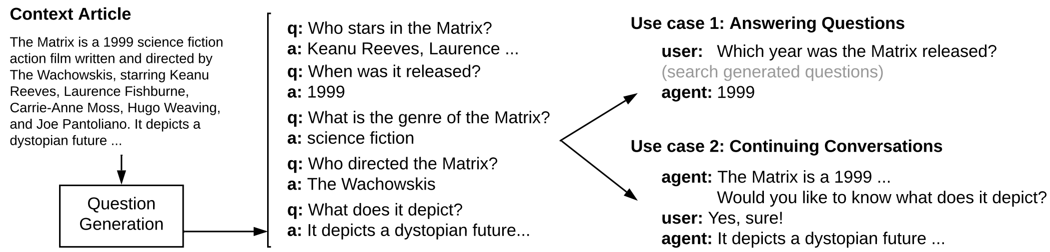


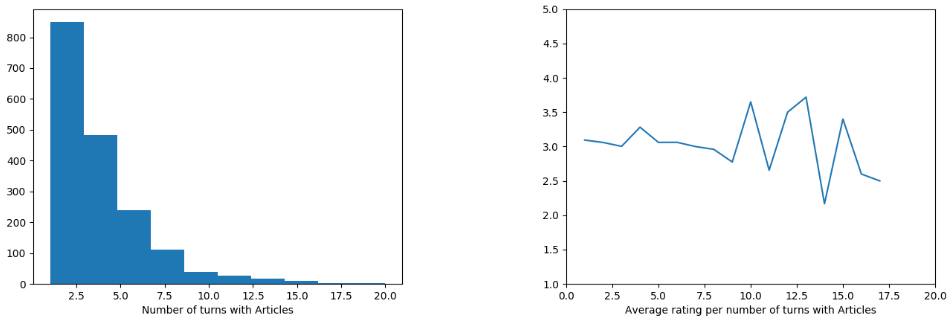
Figure 3: **Question Generation Model** used offline to over-generate questions and answers, facilitating accurate question answering (use case 1) and generation of engaging continuations (use case 2) very cheaply at run time.

Neural Question Generation In natural language processing, reading comprehension (or question answering) has been a prominent area of research, with large data-gathering efforts [Rajpurkar et al., 2016, Reddy et al., 2019, Choi et al., 2018, Dua et al., 2019] and major modeling achievements [Seo et al., 2016, Devlin et al., 2018]. We will build upon more recent work on *question generation*, where the task of question answering has been *flipped* to design neural models that generate question and answer pairs given the context paragraph, where answers are spans or sentences from the context (see Figure 3 left, for example). In particular, we fine-tuned a GPT2 [Radford et al., 2018] to generate questions given context and answer on SQuAD and NewsQA dataset. We generated questions for answer candidates obtained from two sources: all individual context sentence which yields broad questions like "what", "how" and all named entities in the context to yield more granular question like "when", "where". These generated question-answer pairs are further filtered by removing questions that are marked *impossible* to answer by a BERT based model trained on SQuAD2. We utilize such a model to both answer queries and generate continuations, as we will describe next.

Efficiently Answering Queries Although question answering has been a heavily studied area of research that has achieved *near-human* results for simple questions [Liu et al., 2019, Yang et al., 2019, Lan et al., 2019], the underlying models are incredibly memory and computation-intensive, resulting in a slow response even on large GPUs available on AWS. To prevent excessive spending on the hardware, to provide an accurate response in a timely manner, and to simplify the process at execution time, we developed an index-based question answering system. In particular, we over-generated the questions for each sentence in all of the articles in our index (offline, using a sophisticated model), which are then added to an index. At the conversation time, i.e. in response to a user query, we find the generated questions that is most similar to the user's query (using a sentence-embedding based approach), and return the answer to the generated question as our response. Thus at test-time, ZOTBOT is only making fast retrieval (to get the article/paragraph) and nearest neighbor queries (to match the question) to the offline index (similar to the Blacklist module), however since the question generation model is complex and trained on large datasets, the accuracy of question answering remains high. An example illustration of this is "use case 1" in Figure 3.

Natural Continuations Question answering models are useful when the user has a specific query, but to have an engaging, open-ended conversation about an article, we need the conversational agents to be able to read out interesting facts from the article. However, sequentially reading out the sentences from the article makes for a poor user experience since it is unnatural, unengaging, and not interactive. We keep the conversation engaging by constructing continuations such that in addition to responding to the user's previous utterance, we also ask a *leading* follow-up question relevant to the current conversation. We heavily used this guideline even while discussing complex news articles to keep the user involved and subsequently delivering the gist of the article in an appealing manner. To this end, we keep all of the generated question-answer pairs as metadata along with news articles in the ElasticSearch index At each turn, ZOTBOT offers to answer a question that can provide more details into the ongoing topic of discussion. The users can choose to continue or switch to another news article or a different response generator altogether.

Examples The example conversation demonstrates the use of both question generation and answering abilities. Sometimes the generated questions can throw red herrings like "Scientists think the



(a) Histogram of the number of turns

(b) Average rating vs number of turns

Figure 4: **Article-related Conversations:** Analysis of user ratings with increasing number of article-related turns over all conversations in March.

true number of infections may be higher.." because the model is forced to produce a broad question based on every sentence in the passage. A better filtering scheme that optimizes generating a bag of questions rather than disparate questions can improve the relevance of follow-up questions.

User:	*recent news*
ZOTBOT:	Did you read this article : Coronavirus infections predicted to grow exponentially; first death outside China; outbreak becomes political Would you like to know What was the response to the report?
User:	*yes*
ZOTBOT:	The U.S. travel restrictions came in the wake of a report released late last week showing that the virus can be transmitted by someone who has not yet shown symptoms of an infection. Do you want to know What do scientists think is the true number of infections?
User:	*sure*
ZOTBOT:	Scientists suspect the true number of infections may be many times higher than the official count. Would you like to know What did the Indonesia do?
User:	*no what is coronavirus*
ZOTBOT:	The coronavirus, which causes pneumonia-like symptoms, and for which there is no vaccine, was first identified in late December in the Hubei provincial capital of Wuhan.
User:	*how can i protect myself against it*
ZOTBOT:	Sorry I don't know how to respond to that. What would you like to talk about next? I can discuss news, movies and sports

Experiments To analyze the user's continued interest in question generation based long-chain follow-ups we looked at the histogram of how often the users continued conversation before switching to a different topic or exiting. On an average, our module generated 5 good quality, heuristically filtered questions generated per article. Figure 4a shows that users were cooperative and engaged.

However, we found that as the conversations grew longer, the users started engaging in more complex questions and our sentence matching model was not able to handle such complex utterances leading to decrease in the ratings over very long conversations (Figure 4b).

Shortcomings Since question generation module was learned on datasets where the crowd-workers had access to the entire context, the questions have a high amount of lexical similarity with the context. Given that the distribution of conversational language is very different from written language, even when the answer to last question "how can i protect myself against it" is present in the context, the similarity of question embeddings fails to get the right question, "What did the advice say should be done?" to retrieve the answer "Medical advice over the past two weeks has emphasized the need to wear masks to stop transmission through respiratory droplets from the mouth and nose."

6 Neural Response Generator for Backup Responses

As described in previous sections, we have already introduced a lot of features so that our socialbot can say something related to user's utterance. However, the conversations with our socialbot are still

Table 4: Effectiveness of neural response generator

Metrics/Config.	NRG introduced		Knowledge		# context	
	Before	After	None	Provided	7	11
Avg. User Rating	3.09	3.05	**3.09**	2.99	2.93	**3.51**
Avg. # Turns	9.24	9.34	9.81	9.92	10.2	10.2
Avg. Total Conv. Time [sec]	153	**275**	258	**387**	145	**171**
Avg. Bot Conv. Time [sec]	27.9	28.1	31.3	30.0	30.9	34.5
Avg. User Conv. Time [sec]	126	**247**	227	**357**	114	**136**

driven by specific keywords and topics given by users such as news, movies and sports. In case none of our response generators can respond, we use user's utterance as a search keyword for news articles or send the utterance as a query to Amazon EVI. In such a case, these features are used as fallback methods in our system. Not surprisingly, news articles found by the query are usually far from user's interests, and the EVI often ends up being unable to respond to such utterances.

To address the issues, we develop the *neural response generator* (NRG), which is a deep learning model designed to automatically generate a response considering the previous dialog texts. Our goal for the NRG is to generate reasonable responses to user's non-informative utterances such as "i see" and "who knows" until our bot's responses trigger utterances that the bot can handle.

In particular, the neural response generator in ZotBot is a GPT-based model [Radford et al., 2018] used as a remote module deployed on a separate system released by Amazon. Amazon's deployed model was fine-tuned in a TransferTransfo fashion [Wolf et al., 2019] on the Topical-Chat dataset [Gopalakrishnan et al., 2019]. In the training session, each input consists of two instances: dialog context and "knowledge," that was obtained using the ground-truth oracle, as described in Gopalakrishnan et al. [2019]. The corresponding target outputs were candidate responses to the dialog context, where the only one of the candidates is the ground-truth response for each input, and the remaining candidates were randomly selected responses.

Experiments As we introduce the neural response generation (NRG) feature, there are many questions to be addressed such as *a*) how the NRG changes users' experience with our socialbot, *b*) whether or not an additional input, knowledge, helps the NRG produce better responses and *c*) how many previous dialog contexts we should feed to the NRG. Here, we individually perform experiments to address each of the above questions with different experimental periods. *i.e.*, there is no overlap between periods for experiments to discuss *(a)* – *(c)*.

Table 4 shows how each of the configurations affected user experience during the quarterfinals, using average ratings, the average number of turns per conversation, and total conversation time (includes both the time ZotBot and the user spend conversing, as well as the computation time needed for ZotBot to formulate a response). As indicated in the second column, using the NRG as a fallback method increased the total conversation time while neither our bot time nor the average number of turns changed significantly. In other words, the existence of the NRG as a fallback encouraged users to speak more in each turn on average. From the average user ratings before and after we introduced the NRG, however, we can see that it is not enough to introduce the NRG to our socialbot, but we need to tune it to improve the ratings.

In the second configuration, we used the NRG as a fallback method without or with *knowledge*, an optional input used to help the NRG generate a response. For the NRG with knowledge, we used a response from EVI (Section 2.1) as knowledge only when EVI responds to user utterance. As shown in Table 4, knowledge input to the NRG slightly degraded the average rating, though the conversation time increased. We were not able to perform comprehensive hyperparameter tuning in this setting, thus it is possible NRG does produce higher quality generations by leveraging knowledge.

For the first two experiments, we fed at most 7 previous dialog contexts (*i.e.*, 4 utterances and 3 texts from user and ZotBot, respectively). In the third experiment, we set a bigger number, 11, to the maximum number of previous dialog contexts, and compared it to 7 previous contexts. Interestingly, the average user rating was significantly improved from 2.93 to 3.51 by enriched input to the NRG, and it helped users speak little bit more than those in the other batch did.

Examples We provide a number of examples highlighting three different aspects of the NRG.

Using neural response generator as a fallback: Since most of our implemented modules are triggered by specific keywords or phrases, our socialbot will notify users that it cannot respond to user's utterance as the utterance does not trigger any modules. The following example illustrates how our socialbots without/with the NRG respond to such utterances. Here, as the last response, our socialbot fails to respond to "hmm yeah" to which the NRG naturally responds.

ZotBot:	Okay. What did you think about Joker's ending?
User:	*not sure*
ZotBot:	Discussing drama works reminds me of Vertigo. Have you seen it?
User:	*no*
ZotBot:	Okay. Do you have another movie you would like to discuss or should we move onto another topic?
User:	*hmm yeah*
ZotBot:	Sorry, I don't know how to respond to it.
w/ NRG:	do you like joker?

Using an optional knowledge input: An optional input, knowledge, would be helpful especially when previous dialog contexts are not enriched (e.g., at the beginning of the conversation). Taking the following dialog as an example, we show that how different the NRG's response without knowledge (**NRG**) is, comparing to that with knowledge (**NRG+K**). Since there are not many contexts as the user says "Can we talk about Taylor Swift?", knowledge that is given by EVI responding to the user utterance helped the NRG keep itself on track of conversation whereas the response from the NRG without knowledge does not look reasonable.

ZotBot:	What do you like to do in your free time?
User:	*can we talk about taylor swift*
NRG:	oh, yeah. i don't have a netflix account. i have to rely on it to stay informed of things.
NRG+K:	she is a singer who is widely known for her storytelling and her work in the music industry.
K (fact):	(Taylor Alison Swift is an American singer-songwriter. She is known for narrative songs about her personal life, which have received widespread media coverage. At age 14, Swift became the youngest artist signed by the Sony/ATV Music publishing house and, at age 15, she signed her first record deal. Her 2006 eponymous debut album was the longest-charting album of the 2000s in the US.)

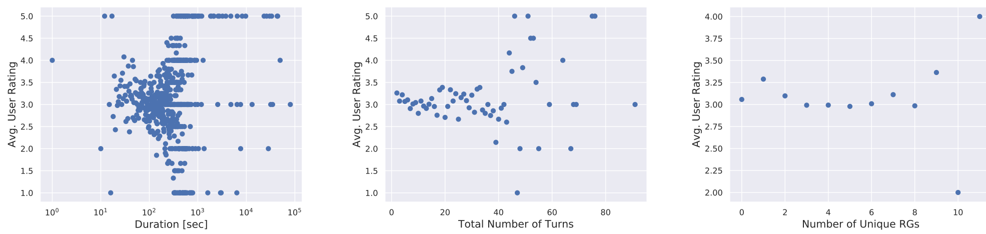
Feeding more previous dialog contexts: As the last example, we provide the following dialog to show different outputs from the NRGs with 7 (**NRG-7**) and 11 (**NRG-11**) previous dialog contexts. With at most 7 previous dialog contexts, the NRG was more likely to repeat specific phrases in its response such as "I don't know. I don't know" than that with at most 11 previous dialog contexts as input. Thus, it would be reasonable to feed more previous dialog contexts as input to the NRG for generating a natural response. In the following example, the NRG repeats "I can." in the last response.

ZotBot:	Do you think you will catch a softball?
User:	*yes i play competitive softball*
ZotBot:	Cool. Well, it has been great discussing softball. Would you like to now discuss movies, news, or something else?
User:	*no thank you*
NRG-7:	i can. i can.
NRG-11:	are you a fan of the sport?

Shortcomings As discussed above, the NRG performs well as a fallback approach. However, there are some important considerations to keep in mind as we use the NRG. First, when rich context is lacking, NRG output is characterized by redundant repetitions in subsequent conversational turns, leading to looping dialogs, or produces responses that ignores the user utterance. See, for example:

User:	*talk to me*
ZotBot:	Hi, this is an Alexa Prize Socialbot. How are you doing today?
User:	*i'm ok*
w/ NRG:	i'm good, thanks for asking.

Second, the outputs can sometimes be inaccurate, offensive, or uncomfortable to users, for example, NRG produced inaccurate responses such as "Alexa prize is the most popular college football team in the world.", and sometimes violated the competition rules (such as generating a name for itself). Third, it is difficult for NRG to maintain long-distance consistency in the dialog, for both facts and opinions, especially when other RGs are also involved. For example, the movie RG informs users



(a) Duration of conversation (b) Number of turns (c) Number of unique RGs

Figure 5: **Average User Rating** in the month of March, plotted against duration, number of turns, and number of unique RGs used in the conversations. Clearly, none of these are significant indicators of when ZotBot provides a satisfactory user experience.

that “Frozen is one of my favorite movies.”, however, if the user expresses that they do not like the movie Frozen, in an attempt to mimic such response, NRG produces contradicting information (e.g., “I didn’t like Frozen.”) in subsequent turns. Lastly, we notice that despite the versatile responses NRG can provide, it has difficulties handling topic transitions, topic maintenance, and conversation breakdowns when user ask for clarification. For example, a conversation may start with engaging back and forth exchanges related to a user’s interested topic on basketball that is maintained by COMeT. When user later requested to talk about his/her favorite food such as “pizza”, NRG still maintains the conversation on the same topic about “basketball”.

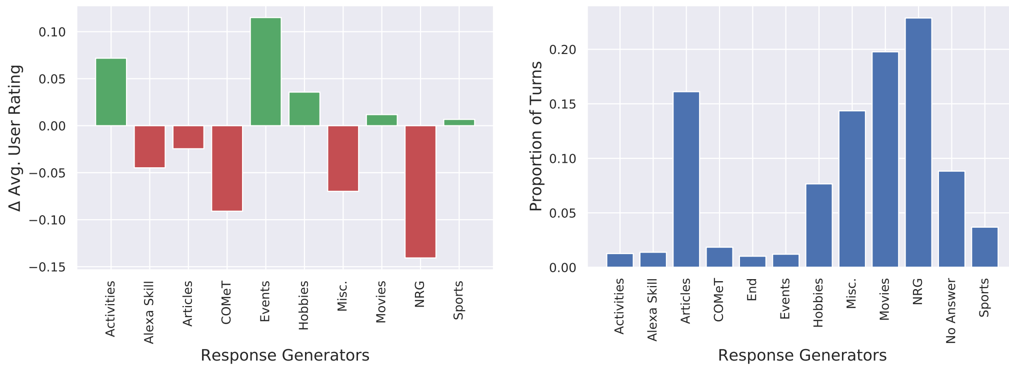
The primary challenge with using NRG is that it is extremely difficult to control the NRG output; since the NRG is an off-the-shelf remote module provided by Amazon, the previous dialog contexts and knowledge are the only parameters to control the output of the NRG.

7 Analysis of the ZotBot Performance

In this section we provide some analysis of the user experience with ZotBot to summarize the performance of the socialbot and the investigation of the different response generators. The analysis in this section is based on approximately 5000 conversations ZotBot had with users in March.

User Ratings Over this period, the average conversation duration was 4:12 minutes, with a median of 1:30, (with an average of 10.5 turns per conversations, a turn defined as one user and one bot utterance), and an average rating of 3.07 (out of 5). We perform some analysis to explore the factors that differentiate good and bad conversations. Figure 5a plots the average user ratings versus the conversation duration, while Figure 5b plots it against the number of turns. With this analysis, it is clear that more turns and longer conversations did not quite lead to better conversations, and thus are not themselves indicative of the problems with ZotBot. To evaluate whether a more dynamic conversation is more indicative of a higher rating, we plot the rating against the number of unique RGs in each conversation (Figure 5c). However, these results are still not good indicators for the quality of the conversation. We now turn to an evaluation that attempts to break down the user ratings over the different response generators involved in the conversation (some of these are introduced in Section 3). Figure 6a shows the average rating for conversations when a response generator is used *at least once* in the conversation, showing higher quality conversations when ZotBot uses the manually-crafted response generators, with a negative effect of the others. NRG shows a significant negative effect here since it is a *fallback* RG, i.e. conversations that use the NRG tend to be low quality ones. There is some variation in the quality of the conversation for different response generators, and thus we investigate the behavior of the response generators.

Response Generators To understand how often each response generator was invoked, we show the proportion of turns that each response generator was used for in Figure 6b (we omit the results in terms of the duration since it provided similar results). Conversing about article using reading comprehension module is quite popular, and used for a significant number of turns. It is clear that even though we develop sophisticated models for commonsense reasoning, the most popular response generators are still manually-scripted ones and the end-to-end neural generation. To better understand



(a) Change over the average rating when active

(b) Proportion of turns for each RG

Figure 6: **Breakdown of Response Generators**: in terms of their effect on the conversation quality, i.e. average ratings when they appeared at least once in (a), and average duration of each response generator, in (b), across all conversations in March

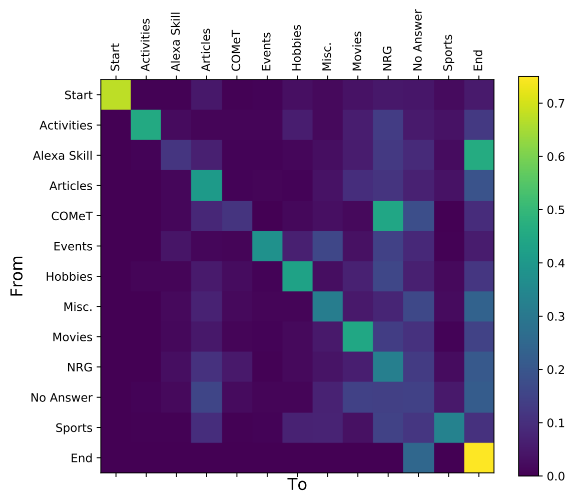


Figure 7: **Transitions between Response Generators**: Over all conversations, we count the number of transitions between the response generators, including *self-transitions*, and plot the probability of outgoing transitions $p(\text{to}|\text{from})$, i.e. each row is normalized.

the dynamics and the dialog flow of ZotBot conversations, we plot the transitions between response generators in Figure 7, normalized for each row (we are computing the conditional probability of transitioning out *from* a given response generator). The high diagonal terms indicate how likely is a response generator to continue its own conversations, with COMeT as a notable RG struggling in this regard as it seems to confuse the user, leading to the backup neural generation taking over (other RGs with this behavior are explicitly single-turn, such as “Alexa Skill” and “No Answer”). User seem to transition to specific response generators as well, such as talking about movies and news articles, but other manually-scripted ones are common destinations as well, and there are no obvious RGs that often end the conversation. Thus even though our NLP-based models generate useful responses (as seen earlier), the users considerably use manually-scripted ones, which may explain our low ratings.

8 Discussion and Lessons Learned

Here we describe some of the major obstacles we faced, and the lessons learned, that may be useful for future participants of the Alexa Prize project, or developers of open-ended conversational bots.

Research Questions Socialbots brings a number of research questions that are substantially different from the focus areas of current natural language processing research [Huang et al., 2020]. One of the most challenging tasks was to maintain an open-domain system that is able to respond appropriately, due to which ZotBot included a number of manually-scripted response generators. Although they were useful to initially bootstrap a competent socialbot, these eventually hindered our progress towards the research tasks. Second, it is extremely challenging for current NLP systems to generate responses that are interesting, correct, and context-appropriate, especially in an open-domain setting, since existing work (i.e. models and datasets) have primarily focused on one or two of these. Third, common sense reasoning is required not only for any socialbot to hold a conversation, but even to give responses that are not completely nonsensical to the users. Our attempt at this was able to generate single, useful responses, but did not lead to multiple interesting turns. This leads to the final challenge, the goal of ensuring that the conversation is coherent across multiple turns and topics, and across multiple domains. Given these challenges, perhaps future of research in socialbots resides not in solving each of these problems, but primarily in using massive neural models for end-to-end generation [Vinyals and Le, 2015], trained on massive datasets [Adiwardana et al., 2020, Stephen Roller, 2020].

Engineering Design For the majority of the competition duration, the research efforts were dwarfed by the effort needed for the engineering tasks, such as hard-coding, setting up/tuning infrastructure, and deployment. This was a significant problem for our team (we focused on an interdisciplinary team consisting of many non-CS majors), and is useful to keep in mind, especially for those participating in the competition for the first time. Focusing on engineering issues here, we share several lessons learned through this competition so that those interested in participating in the future competitions can avoid the same mistakes.

Coding style agreement: Needless to say, in such a large-scale project it is critical to write code readable (and understandable) to other members as you share code repositories. Otherwise, such technical debts will accumulate as the competition goes on, and no one else can maintain/debug the code. From our experience, we strongly recommend teams discuss their coding style and decide it at the very beginning of the competition, such as picking a standard style guide.

Code review system: Similarly, it would be important to discuss and decide how the code should be reviewed before deployment. For instance, the format of pull requests is also something to decide before using the code review system. The reviewer should not only test the functionalities but also check the coding style. To reduce the workload for code review, we would recommend use of a hosted continuous integration and deployment system such as Travis CI as part of development.

User experience review beyond text: One of the core metrics to assess socialbots in this competition is an average of user ratings that are given at the end of conversations. To see how the conversations with our socialbot had impacted the user ratings, we analyzed dialogue as sequences of texts until the quarterfinal round. As described in the previous sections, we introduced a lot of features/modules to cover various topics and make the conversation flow smoother, and the improvements could be confirmed through the transcripts. This, however, turned out to be a critical mistake; our analysis was a little bit too much focused on the dialog texts and missed response time, a critical aspect in conversations with users. As we significantly updated ZotBot, the utterances improved, but some of our complex modules (including remote ones) caused significant delays (*e.g.*, sometimes it took up to 9 seconds to respond to simple utterances) that degraded user experiences as a result.

Fine-grained log analysis system: Introducing fine-grained log analysis was key to a better understanding ZotBot's behavior, such as in the A/B testing. As for delay analysis, it would be critical to review not only overall conversation time but also the duration for user-and-bot-sides and module-level delays. To perform such log analyses in automatic ways, defining well-formatted log outputs is essential. From our experience, it is worth having at least one member be an expert in, and devoted to, log analysis on a daily basis.

User Experience From reviewing user transcripts over the course of the competition, we identified several lessons to inform better user-centered conversational design of social chatbots.

Anticipate User Response: Prior to creating manually crafted RGs, conversation experience designers and engineers should work collaboratively to define anticipated user responses. Doing so will allow designers and developers to carefully consider how and where to use open-ended questions (e.g., what, where, who) to probe speaker interest and close-ended confirmatory questions (e.g., Do you like <movie name>? Is it <movie> one of your favorites?) [McTear, 2018].

Latency in RGs: Response time is another factor that can impact the quality of conversations, as we noticed an increase in user frustration, and terminations, with increased delays in responses. The neural response generator had one of the highest latencies as it relied on using an external API, in addition to the necessary inference time. The Movies response generator also had a longer delay, as multiple AIQL calls were made to find relations between named entities. To reduce the duration of some response generators like activities-based, information such as attributes was indexed offline and stored in a DynamoDb table. It should be highlighted that the distribution of response generators varies significantly, as depicted in Figure 6b.

Manage Communication Breakdowns: In cases that RGs fail to respond to user request, a fall-back handler (e.g., NRG) should be implemented with diversified conversation prompts to redirect users back to topics and keywords that known RGs can handle [Pearl, 2016]. No response conditions (e.g., "none, not interested") should always be considered for all RGs, with the help and support from iterative and rigorous internal user testing.

Transition Across RGs: As multiple RGs are being created asynchronously by different developers with diverse levels of experience and skills, it is critical to evaluate whether different RGs can redirect user utterances that are difficult to handle to other RGs. This prevention mechanism can minimize conversation breakdowns and also ensure overall dialog flow experience for users.

Filter Inappropriate Content: Many RGs that rely on previously trained data may encounter difficulties providing context-appropriate responses, therefore, whitelisting and blacklisting should be constantly updated with content filtering for both user utterances as well as the bot's own utterances.

Building Bot Specific Persona: In addition to using third-party knowledge-base for RGs, we recommend personalizing RGs with user-generated knowledge and implementing a persona-model [Nguyen et al., 2017] to handle user inquiry questions, especially to inform a bot's unique characteristics (e.g., favorite songs, books, and shows).

New Users: Users may have diverse levels of experiences in initiating and terminating bot conversations using distinct keywords, these idiosyncratic patterns of use can accidentally trigger the social bot and/or causing user challenges in finding the right verbal commands to terminate conversations. The addition of a tutorial for new users can be helpful to introduce new users to the concept of a social bot, and clarifying the distinction between a social bot and another software application can reduce user frustration. Adding conversational fillers to show emotional support [Wang et al., 2020] and inform users to use specific commands can provide appropriate guidance to novice users during the conversation.

9 Conclusions

We described the ZOTBOT system in this report, including the system design and the manually-scripted response generators, along with the use of sophisticated NLP models for common sense reasoning, reading comprehension, and backup responses. We presented numerous examples and different analysis to highlight the main advantages of ZOTBOT, along with identifying key challenges and lessons for future work in open-domain conversational agents. Some of the gathered datasets and source code, along with further information about the team, is available here: `https://uci-zotbot.github.io/`.

Acknowledgements

We are grateful to Amazon for supporting this work, both financially and technically. We would like to thank Claire Urata, Jeanette Mai, Michelle Liang, Xuan Liu, and Moeez Qamar for their valuable input and analysis of ZOTBOT; they were a crucial part of the development pipeline and ZOTBOT would not have been the same without them. We would also like to thank Matt Gardner and Magdalena Nitchi for their help with the application and thank Juan Garcia for providing his insightful football analysis.

References

Daniel Adiwardana, Minh-Thang Luong, David R. So, Jamie Hall, Noah Fiedel, Romal Thoppilan, Zi Yang, Apoorv Kulshreshtha, Gaurav Nemade, Yifeng Lu, and Quoc V. Le. Towards a human-like open-domain chatbot, 2020.

Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. Comet: Commonsense transformers for knowledge graph construction. In *Association for Computational Linguistics (ACL)*, 2019.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*, 2018.

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

Yun-Nung Chen, Tania Bedrax-Weiss, Dilek Hakkani-Tur, Anuj Kumar, Mike Lewis, Thang-Minh Luong, Pei-Hao Su, and Tsung-Hsien Wen. Proceedings of the first workshop on nlp for conversational ai. In *Proceedings of the First Workshop on NLP for Conversational AI*, 2019.

Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wen-tau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. Quac: Question answering in context. *arXiv preprint arXiv:1808.07036*, 2018.

Kevin Clark and Christopher D Manning. Deep reinforcement learning for mention-ranking coreference models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2256–2262, 2016.

Henry K Dambanemuya and Nicholas Diakopoulos. "alexa, what is going on with the impeachment?" evaluating smart speakers for news quality.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. *arXiv preprint arXiv:1903.00161*, 2019.

Jianfeng Gao, Michel Galley, Lihong Li, et al. Neural approaches to conversational ai. *Foundations and Trends® in Information Retrieval*, 13(2-3):127–298, 2019.

Karthik Gopalakrishnan, Behnam Hedayatnia, Qinlang Chen, Anna Gottardi, Sanjeev Kwatra, Anu Venkatesh, Raefer Gabriel, Dilek Hakkani-Tür, and Amazon Alexa AI. Topical-chat: Towards knowledge-grounded open-domain conversations. *Interspeech*, 2019.

Minlie Huang, Xiaoyan Zhu, and Jianfeng Gao. Challenges in building intelligent open-domain dialog systems. *ACM Trans. Inf. Syst.*, 38(3), April 2020.

Khatri, Hedayatnia, and Venkatesh et. al. Advancing the state of the art in open domain dialog systems through the alexa prize. *ArXiv*, abs/1812.10757, 2018a.

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

Chandra Khatri, Anu Venkatesh, Behnam Hedayatnia, Raefer Gabriel, Ashwin Ram, and Rohit Prasad. Alexa prize—state of the art in conversational ai. *AI Magazine*, 39(3):40–55, 2018c.

Thomas Kollar, Danielle Berry, Lauren Stuart, Karolina Owczarzak, Tagyoung Chung, Lambert Mathias, Michael Kayser, Bradford Snow, and Spyros Matsoukas. The alexa meaning representation language. In *North American Chapter of the Association for Computational Linguistics: Industry Papers (NAACL)*, pages 177–184, 2018.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

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

Michael McTear. Conversation modelling for chatbots: current approaches and future directions. *Studientexte zur Sprachkommunikation: Elektronische Sprachsignalverarbeitung*, 2018.

Vincent Ng. Supervised noun phrase coreference research: The first fifteen years. In *Association for Computational Linguistics (ACL)*, 2010.

Huyen Nguyen, David Morales, and Tessera Chin. A neural chatbot with personality, 2017.

Peter Norvig. Unified theory of inference for text understanding. Technical report, CALIFORNIA UNIV BERKELEY GRADUATE DIV, 1986.

Cathy Pearl. *Designing voice user interfaces: principles of conversational experiences*. " O'Reilly Media, Inc.", 2016.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf*, 2018.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

Abhinav Rastogi, Alexandros Papangelis, Rahul Goel, and Chandra Khatri. Wcis 2019: 1st workshop on conversational interaction systems. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1435–1437, 2019.

Siva Reddy, Danqi Chen, and Christopher D Manning. Coqa: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266, 2019.

Maarten Sap, Ronan Le Bras, Emily Allaway, Chandra Bhagavatula, Nicholas Lourie, Hannah Rashkin, Brendan Roof, Noah A. Smith, and Yejin Choi. Atomic: An atlas of machine commonsense for if-then reasoning. *ArXiv*, abs/1811.00146, 2019.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*, 2016.

Robyn Speer, Joshua Chin, and Catherine Havasi. Conceptnet 5.5: An open multilingual graph of general knowledge. In *AAAI Conference on Artificial Intelligence*, 2017.

Naman Goyal Da Ju Mary Williamson Yinhan Liu Jing Xu Myle Ott Kurt Shuster Eric M. Smith Y-Lan Boureau Jason Weston Stephen Roller, Emily Dinan. Recipes for building an open-domain chatbot. 2020.

Oriol Vinyals and Quoc V. Le. A neural conversational model. *CoRR*, abs/1506.05869, 2015.

Jinping Wang, Hyun Yang, Ruosi Shao, Saeed Abdullah, and S. Shyam Sundar. Alexa as coach: Leveraging smart speakers to build social agents that reduce public speaking anxiety. In *CHI Conference on Human Factors in Computing Systems*, 2020.

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

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5754–5764, 2019.

Koichiro Yoshino and Tatsuya Kawahara. Conversational system for information navigation based on pomdp with user focus tracking. *Computer Speech & Language*, 34(1):275–291, 2015.

Tom Young, Erik Cambria, Iti Chaturvedi, Hao Zhou, Subham Biswas, and Minlie Huang. Augmenting end-to-end dialogue systems with commonsense knowledge. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.