
Proto: A Neural Cocktail for Generating Appealing Conversations

Sougata Saha^{*}, Souvik Das^{*}, Elizabeth Soper[†], Erin Pacquetet[‡], Rohini K. Srihari[‡]

Department of Computer Science & Engineering, Department of Linguistics
University at Buffalo
Buffalo, NY 14260

{sougatas, souvikda, esoper, erinmorr, rohini}@buffalo.edu

Abstract

In this paper, we present our Alexa Prize Grand Challenge 4 socialbot: Proto. Leveraging diverse sources of world knowledge, and powered by a suite of neural and rule-based natural language understanding modules, state-of-the-art neural generators, novel state-based deterministic generators, an ensemble of neural re-rankers, a robust post-processing algorithm, and an efficient overall conversation strategy, Proto strives to be able to converse coherently about a diverse range of topics of interest to humans, and provide a memorable experience to the user. In this paper we dissect and analyze the different components and conversation strategies implemented by our socialbot, which enables us to generate colloquial, empathetic, engaging, self-rectifying, factually correct, and on-topic response, which has helped us achieve consistent scores throughout the competition.

1 Introduction

Expressing thoughts and interacting with one another using dialogue is instinctive for humans. Hence, creating an artificially intelligent socialbot that can converse with a human, at par with human-level conversational capability, is very difficult. Proto is our attempt to create a socialbot that exhibits human-level conversation competency, and engages in conversation with humans, on any permissible topic. With a state-of-the-art ensemble of neural response generators, which are powered by large scale knowledge bases like Wikidata, and leveraging other latest research in natural language processing and conversational AI, Proto is an end-to-end conversational system which can not only understand and generate responses, but also drive the conversation deeper, be it any topic. This paper introduces and describes in detail how our Alexa Prize 2020 socialbot is able to achieve consistently high scores in conversations with real users.

A socialbot is generally defined as an artificially intelligent agent, that can participate in an open domain conversation with a human user on any topic, much like talking to a stranger at a bar. Unlike task or goal-oriented systems, where the bot aims to perform certain actions like successful ticket reservation, customer support, etc. the unrestricted domain of socialbots, has no well-defined objectives other than engaging in interesting conversations. This introduces novel and exciting challenges, which demand multi-disciplinary solutions.

While keeping Grice's Maxims in mind, several considerations need to be accounted for, in order to generate an appropriate and succinct response to a user query. We broadly identified and incorporated

^{*}Equal Contribution and team leads. Department of Computer Science & Engineering.

[†]Authors are in alphabetical order. Department of Linguistics.

[‡]Faculty advisor. Department of Computer Science & Engineering.

the following technologies and strategies in our system, in order to handle the wide range of user needs.

- **Natural Language Understanding:** The ability to understand and extract features (intent, sentiment, entities, etc.) from the user’s query. In order to generate an appropriate response to a user query, it is imperative to understand and extract features from the query, which can be acted upon by the response generators.
- **Curated Content Selection and Delivery:** The ability to select and incorporate relevant factual content (encyclopedic knowledge) and interesting content (fun facts) in order to generate a response. To be able to generate factual, interactive, and interesting responses, it is important to efficiently collect, process, and store disparate knowledge sources offline, and efficiently retrieve and utilize them with minimum latency online. Leveraging Elasticsearch indexes containing Wikidata and scraped interesting facts, we are able to increase the topic coverage of the bot.
- **Colloquialism:** The ability to generate responses in casual (day-to-day) language. During the competition, we observed that template-based handwritten response generally lack the naturalness and colloquialism which is required by a bot to sound like a human, we also observed that distinguishing natural responses from template-based handwritten responses is a relatively easy task for humans, which can give the perception of the bot not actually understanding the user, and making the user skeptical about the bot. Our ensemble of neural generators trained on diverse human-to-human conversations helps alleviate this problem, and makes the conversation more natural, and increases the humanness of the bot.
- **Remembering User Attributes:** The ability to retain and reuse specific pieces of information shared by the user (example: name, favorites, etc.). Remembering and using personal information generally helps make the conversation more colloquial and cordial. The natural language understanding module extracts slots from the user’s query, which contain the user’s name and preferences around specific topics (favorite movie, food, etc.), which is used throughout the conversation.
- **Exhibiting Emotions:** The ability to understand user emotions and respond emphatically. As humans are emotional creatures, emotionally adept responses help bridge the barrier between the bot and the user. We noticed that many times the users seek emotional support from the bot and share their personal encounters with the bot, and an empathetic response to such user queries generally leads to a better user experience, thus resulting in better approval ratings. Leveraging a suite of neural generators trained on human-to-human empathetic conversations, complimented by hand-written scenario-based empathetic responses, our bot is able to demonstrate empathy in its responses.
- **Diving Deep:** The ability to dive deep into a topic, and engage in logical exchanges with the user. Human conversations are generally multi-layered in nature. For example, when humans discuss the theme ‘movies’, the initial layers might consist of talking about liking or disliking movies in general, with the further layers discussing the reason for the same, followed by discussing specific movies and specific scenes from the movies which created an impression. Also, in daily conversations, switching to a different, related theme on encountering certain trigger words or phrases is common. For example, while discussing cars, switching the topic to the movie ‘Fast and Furious’ seems logical and natural. We tried to mimic this behavior of diving deep into conversations and naturally transitioning to related themes in our bot, through our deterministic and neural generators. A robust retrieval and re-ranking-based prompt selection module helps us dive deeper into scenarios where the generators are incapable of advancing the conversation, and also enables the bot to transition to a logical next theme, based on certain words or phrases in the context of the current discussion. We found this to be one of the most difficult criteria to tackle, but is crucial for achieving truly human-like conversation.
- **Maneuverability:** The ease of navigating and eliciting response or reaction from the bot. Just like driving a car, the user should be able to steer the bot with ease and should be able to evoke a reaction or response from the bot using dialogue. For example, they should be able to change the current topic of discussion, get a response to queries (empathetic, opinions, etc.), go back to a previous topic of discussion, suggest topics or entities to discuss, express dissatisfaction, and end the conversation when desired. Our suite of natural

language understanding modules helps us understand the intent of the user better, which aids in generating a desirable response. Throughout the competition, we noticed a positive correlation between high user ratings and increased maneuverability of the bot.

- **Programmatic Change of Control:** The ability to change the narrative of the conversation by switching the control of the conversation between the user and the bot, as and when required. Generally in human-to-human conversations, both actors are responsible for advancing the conversation. However, in human-to-bot conversations, we noticed that users, in general, don't help much in progressing the conversation, and resort to bland and short responses. Hence, the onus is on the bot to advance the conversation, which is best done by asking questions to the user. However, a drawback of this strategy is that the user experience might be hurt due to questions in every turn. Hence, we not only implement strategies to prevent prompts in every turn thus providing control to the user but also encourage the user to introduce new themes and entities in the conversation, in a controlled way.
- **Self-Rectification:** The ability to recover from an undesired state in the conversation. Sometimes the bot might misunderstand the intent of the user, hence enter a state which is inappropriate at that point in time, and generate misleading responses. In the future turns, this problem might cascade, and take the conversation in a different direction altogether. To recover from these states, we actively ask feedback from the user every k turns, to check if they are interested in continuing with the current theme or want to switch to a different topic. Our natural language understanding module is also able to detect dissatisfaction, negative intent, and complaints from the user, which we can then respond to with an appropriate message, and transition the discussion to a different theme.

Having given a high-level overview of the features of a successful socialbot, in the following sections we discuss how we have realized these features in detail, and then analyze the impact of the features on users' experience with the bot.

2 End-to-End Flow: Walk-through of response generation

Figure 1 illustrates our end-to-end architecture at a high level. The journey of a user query inside our bot starts when the Amazon Speech-to-Text model converts user speech into text and passes the query to our bot. Upon receiving a text query, the dialog manager invokes a suite of Natural Language Understanding (NLU) modules, which extract diverse features like user sentiment, entity mentions, conversation topic & theme, user current satisfaction, user intent, diverse slot values .etc. Post feature extraction, the dialogue manager links and tracks the entity mentions (if any) to Wikidata, and invokes an appropriate set of response generators in parallel, in order to generate a set of candidate responses to the user query. Once the candidate responses are generated, an ensemble of neural re-rankers utilizes the current query and the conversation context for ranking and selecting the best response. A post-processing module enhances the best response by adding prompts, salutations .etc, or removing contradictions, excess punctuation, phrases .etc. The response from the post-processing module is returned to Amazon's Text-to-Speech model, which generates and delivers the audio response to the user. The dialog manager saves the state of the current conversation in a DynamoDB state table, which is reused in the next turn. In the next sections, we explain each of the components in detail, which enables us to generate an accurate response in minimum latency.

3 Natural language understanding modules

The natural language understanding modules extract several features from the utterance, which are essential for the different downstream modules. The outputs of the NLU modules are stored in the state manager of the conversation, which gets written into DynamoDB.

3.1 Query Punctuation, Named Entity Recognition & Sentiment Detection:

Since the text provided by the ASR module does not contain any punctuation, we leverage the BERT-based punctuation model present in the Cobot environment for re-punctuating the user query.

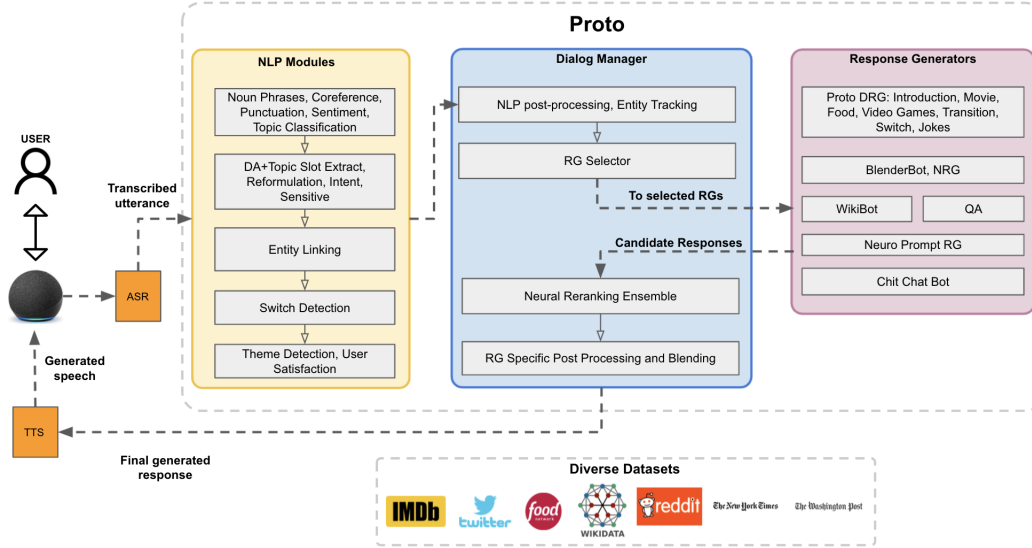


Figure 1: Proto System Architecture

| Name | Example | # Training Examples |
|------------------------|--|---------------------|
| acknowledgement | <i>Yes</i> | 165 |
| rejection | <i>No</i> | 92 |
| clarification | <i>What?, I didn't understand that.</i> | 80 |
| state_personal_fact | <i>I have a dog.</i> | 1093 |
| state_knowledge_fact | <i>Water boils at 100 degrees celsius.</i> | 1111 |
| state_opinion | <i>I like this film.</i> | 1797 |
| request_personal_fact | <i>Do you have a dog?</i> | 571 |
| request_knowledge_fact | <i>How old is George Clooney?</i> | 297 |
| request_opinion | <i>Do you like comedies?</i> | 298 |
| topic_suggestion | <i>let's talk about movies.</i> | 31 |
| general_chat | <i>How are you?</i> | 486 |
| other | Anything other than the above classes | 171 |

Table 1: Types of User Intent

Detecting the entities from the user text is a crucial task. Accurate entity extraction not only enables our bot to stick to the topic the user is interested in but also helps us retrieve relevant factual information regarding the entity of interest. We leverage the BERT-based NER model extended in Cobot for extracting entities from the user query.

Along with determining the entities, it is important to understand the sentiment of the user at any given point of the conversation. For our purpose, we leverage the lightweight VADER (Valence Aware Dictionary and Sentiment Reasoner) sentiment analyzer and restrict the sentiment classes to be positive, negative, or neutral, along with the intensity of the sentiment, ranging between -1 to 1. Efficient sentiment detection enables us to generate emotionally appropriate responses and enhances maneuverability across topics, thus providing an engaging experience to the user.

3.2 User Intent Detection

Human conversations exhibit certain patterns. For example, when someone asks a personal question, the response is generally a personal statement or an opinion. When a piece of information is requested, the response is generally a factual statement. In order to leverage such patterns, it is imperative to understand the intent of a user query. Analyzing existing conversation datasets, we define 12 intent categories that commonly exist in a conversation. Table 1 details the defined intent classes. We annotated a subset of Topical Chat dataset (1), Chit-Chat dataset (2), and ConvAI-2 dataset (3) to

create training examples for a BERT-based (4) intent classifier, which is hosted outside of the Cobot environment, and accessible using API calls. Table 2 details the model’s performance on the test set.

| Model | F1 | Accuracy |
|-------------------|-------|----------|
| bert-base-uncased | 71.48 | 71.7 |

Table 2: BERT-based intent classifier metrics on test set

| Model | F1 | Accuracy |
|-------------------|-------|----------|
| bert-base-uncased | 97.78 | 97.74 |

Table 3: BERT-based sensitive classifier metrics on test set

Unfortunately, many times users try to elicit a reaction from the bot by using foul and sensitive language. In order to detect such queries, we implement an iterative approach. In the first iteration, we leverage the high precision, keyword-based sensitive classifier developed by Stanford Chirpy Cardinal (5). In order to increase the recall, in the second iteration, we leverage a BERT-based sensitive query detection classifier, which is only invoked if the text is not classified as sensitive in the first iteration. For training the BERT-based sensitive query classifier, we annotated a subset of the CAPC dataset (shared by Amazon), and the Bot-Adversarial Dialogue dataset (6). The combined dataset contains 24,250 non sensitive examples and 21,869 sensitive examples. Table 3 details the model’s performance on the test set.

3.3 Slot Based Information Extraction

In order to extract actionable information from a user query, we analyzed a subset of existing conversations with our bot and developed a high-precision regex-based slot extraction module. The module can extract the user’s name, age, relationship mentions, liking, disliking, wanting, and intent to switch the discussion to a different topic or entity. The extracted information is generally used by downstream modules, and certain slots like user name, user favorites, etc. are persisted and reused throughout the conversation.

3.4 Query Reformulation

Colloquial human conversations are rife with co-reference, making it natural for users to use anaphora while interacting with our socialbot, which makes the task of understanding user intent challenging. Leveraging the conversation context, we implement an ensemble of Spacy neuralcoref (7; 8), and Allen AI co-reference resolution (9) to reformulate the latest user query and make it more adequate with information. We use Allen AI’s co-reference resolution as the primary resolver but resort to Spacy if it fails to resolve the co-references.

3.5 Entity Linker

To understand precisely which entities a user is talking about in a conversation, Entity Linking plays an important role. This module is extensively used in several areas in our system like response generation, topic understanding, entity tracking, etc. To make the system scalable the Entity Linking engine is decoupled from the Cobot framework and hosted in an EC2 instance. Along with our key changes in scoring strategy to address the conversational context and rarity of the spoken entity, our approach was inspired in large part by last year’s Chirpy Cardinal’s open-source release(5).

We processed a dump of English language Wikipedia. The following information for an entity (referred as E) is collected (a) *pageviews* (number of pageviews in a month) (b) the *anchortext distribution* $P_{anchortext}(a|E)$

The anchortext distribution for an entity E is calculated using the following formula:

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

Here, we count the fuzzy matches between *anchortexts* (lowercase) with the hyperlinks related to the entity E . The python library *fuzzywuzzy* is used to get the token similarity, the similarity threshold is kept at 0.95. $A(E)$ is a set of all anchortext for the entity E .

The user utterance u is segmented in n-grams where $n = 5$ and stored into a set S . After that, an Elastic Search query is constructed to fetch all entities E which has at least one span $s \in S$. S is

used to constitute the search keywords, where stop words are neglected and a boost parameter is provided where the uni-gram frequency of the set element is less than 10. To link the entity we estimate $P(E|s)$, i.e. likelihood of a span s is referring to an entity E , $P(E|s)$ can be estimated using a Bayesian model:

$$P(E|s) \propto P(E) \times P(s|E) \quad (2)$$

Here, $P(E)$ is assumed to be equal to *pageviews* and $P(s|E) = P_{anchortext}(s|E)$. Also, we introduce a rarity coefficient α which factors in the rarity of occurrence of an entity and θ as contextual coefficient which accounts for the context of the entity in a conversation. Therefore, the *score*(s, E) between the span s and entity E can be defined as:

$$score(s, E) = \alpha \times \theta \times P(E) \times P_{anchortext}(s|E) \quad (3)$$

$$\alpha = \frac{1}{\text{unigram frequency}(s) + 1} \quad (4)$$

For calculating the contextual coefficient, we concatenate the last two utterances and the current utterance into a string joined by [SEP] token, this is referred to as *context*. Also, each entity E is represented by the top three items in the categories field obtained from the Wikipedia dump. The three strings are concatenated into a string joined by [SEP] token, this is referred to as *c_{representation}*. Now the context and candidate representations are passed through a cross-encoder which scores the entities based on the context. The raw scores are then normalized and we discard entities that have scores below a threshold. θ is defined as follows:

$$\theta = \text{NORMALIZE}(\text{CrossEncoder}(\text{context}, c_{\text{representation}})) \quad (5)$$

For ranking the entities we follow the same strategies as used by previous year’s Chirpy Cardinal’s system.

3.6 Entity level Auxiliary Data

While processing the Wikipedia dump, we also process some auxiliary data for all entities which are used by the response generators: (1) **Summary sentences:** We are using LexRank(10) summarizer to get top-5 sentences from the overview section of an entity, these sentences are treated as facts by the neural generators (2) **Key-phrases:** We are also extracting top-10 key-phrases from the overview section using Rake(11). These key phrases are used to contextually switch to a related entity if the response generator hits a roadblock.

3.7 Current topic & Theme Detection

In order to generate a relevant response and guide the conversation better, it is important to understand the topic of a user utterance. However, an utterance can belong to multiple topics. For example, the statement "I was playing Fifa while eating a bag of chips" contains references to both video games and food. Also, there are scenarios where a user tries to unnaturally drift to another topic. For example, when asked "What is your favorite movie ?", a user can respond with "I ate burgers for dinner today. " Although unnatural, we have observed such scenarios, hence making it important to efficiently detect and track the topic. In order to tackle these unnatural topic changes, we introduce the concept of theme, where we constrain a single utterance to belong to only one topic. We employ an ensemble of the Cobot environment’s inbuilt topic classifier, entity linker’s predicted topic, and a RoBERTa-based classifier (12) trained on the ConCET dataset (13) to detect the most probable topic clusters of the current utterance. The predicted topic cluster from the current query, along with the theme of the previous turn, helps us decide if the current theme of the conversation needs to be changed. The RoBERTa based classifier is trained to classify a sentence into 1 of 17 classes: attraction, celebrities, chitchat, fashion, fitness, food, games, joke, literature, movie, music, news, other, pets animals, sports, tech, and weather. The classifier achieves an accuracy of 0.97, and macro F1 score of 0.94 on a held-out test set. Table 4 details the classification metrics achieved for each of the themes, on the test set.

| Class | Precision | Recall | F1 | Support | Training examples |
|--------------|-----------|--------|------|---------|-------------------|
| attraction | 0.95 | 0.95 | 0.95 | 687 | 5,949 |
| celebrities | 0.89 | 0.89 | 0.89 | 168 | 1,575 |
| chitchat | 0.99 | 0.99 | 0.99 | 4,331 | 39,401 |
| fashion | 0.95 | 0.99 | 0.97 | 74 | 658 |
| fitness | 1.00 | 0.92 | 0.96 | 12 | 102 |
| food | 0.98 | 0.98 | 0.98 | 334 | 3,205 |
| games | 0.92 | 0.69 | 0.79 | 16 | 169 |
| joke | 0.96 | 1.00 | 0.98 | 54 | 486 |
| literature | 0.99 | 0.99 | 0.99 | 180 | 1,651 |
| movie | 0.97 | 0.98 | 0.98 | 16,645 | 149,283 |
| music | 0.99 | 0.98 | 0.98 | 2,963 | 26,636 |
| news | 0.95 | 0.94 | 0.94 | 6,339 | 57,807 |
| other | 0.93 | 0.98 | 0.95 | 645 | 5,361 |
| pets animals | 0.97 | 0.92 | 0.95 | 250 | 1,939 |
| sports | 0.96 | 0.94 | 0.95 | 2,159 | 18,931 |
| tech | 0.97 | 0.96 | 0.97 | 2,213 | 20,469 |
| weather | 0.90 | 0.82 | 0.86 | 66 | 602 |

Table 4: Theme classifier performance on the test dataset.

3.8 User Satisfaction Detection

Detecting the level of user satisfaction with the current theme and bot’s responses, and acting accordingly is crucial for delivering a good user experience. Inspired by Chirpy Cardinal’s satisfaction detection module, and complemented with our own suite of detectors, we implement a set of regex-based detectors, which can detect user complaints and level of engagement, and help generate better responses, or redirect the conversation to a more relevant theme if needed.

3.9 Fact Retrieval

Interesting facts make a conversation more lively and engaging. We collated and indexed fun facts by scrapping online resources and complemented them with Topical Chat fun facts, to create a corpus of 30,000 interesting facts. Given the current user query, we perform a semantic search using a pre-trained bi-encoder (14) to retrieve the top fun facts and Wikidata facts (if any entity is detected by the entity linker), and re-rank the facts using a pre-trained cross-encoder (14) to select the top k relevant facts given the query. The retrieved facts are used by the neural generators to generate an interesting response.

4 Dialogue Manager

The dialogue manager is the core module that is responsible for generating a response. It keeps track of the entities in the current conversation, and invokes the appropriate set of response generators, using the extracted features from NLU. Below we discuss the two main components of the dialogue manager in detail.

4.1 Entity Tracking

Entity Tracking is one of our core components that ensures our bot is talking about the entities that are introduced by the user. Different response generators consume the Entity Tracking data to work in tandem and answer user queries. The main part of our entity tracker is a set of caches, which (i) stores the entities which we have already been discussed in the past, (ii) stores the current set of entities that are currently being discussed, and (iii) tracks a set of entities which were mentioned by the user, but not discussed in the past. Below we explain in detail the life-cycle of an entity, in our Entity Tracking module:

1. **Initialize Entity Tracking Cache:** At the start of every conversation we initialize a cache that stores the current and previous entities, the entities which are rejected by the user or that the user is unsatisfied with, the entities that we have finished talking about, and the entities which can be talked about in the future.

2. **Detect Navigational Intent:** After the cache is set up we analyze the user utterance. Firstly, we detect the navigational intent of the user, i.e. to check the user wants to talk about a particular entity or expresses disinterest in it. To detect navigational intent we have used some of the regexes released by Chirpy Cardinal and complemented that with our own regexes.
3. **Current Entity Detection:** If the user expresses positive navigational intent about an entity and the Entity Linker can link that entity then we assign that entity to the current entity. Otherwise, if negative intent is expressed the entity is placed into the rejected list. Users can refer to an entity multiple times, for that reason we do co-reference resolution in the subsequent turns after a new entity is discovered and check with the previous entity. If no new entity is found the old entity is preserved unless a topic switch is detected.
4. **Entity Recommendation:** While a majority of the time our Entity Linker links an entity properly, there are cases where an entity of the expected type is not found or simply the spoken entity is not present in Wikipedia. In these cases, we recommend an Entity that was introduced by the user in the earlier utterances but was not spoken about. Here we select a most recent Entity, we also check if the user has expressed dissatisfaction with any other entity of the same type.

4.2 Response Generation Strategies & Generator Selection

Throughout the competition, we experimented with multiple response generation strategies and found the combination of the below strategies to perform best.

1. **Launch:** Always invokes the Launch RG, which welcomes the user by introducing itself with the statement ‘Hi, this is an Alexa Prize socialbot.’, followed by asking the user how they are doing.
2. **Initial / Greeting Flow:** Post-launch state, we experimented with multiple greeting/introduction flows: (i) Covid-19 intro: The Covid intro engages in empathetic discussion around the current Covid-19 pandemic, and the vaccines devised to tackle the virus. Until the Quarterfinals, the Covid introduction yielded consistent user ratings. However, as the Covid-19 situation eased in the US, we noticed reluctance among users to talk about Covid and vaccines, which resulted in a decline in ratings. This flow has been decommissioned since the semi-final phase. (ii) Upcoming/Latest events intro: This generic introduction flow can be configured to discuss a few turns around any latest or upcoming events. For example, two weeks before the 2021 Academy awards, we introduced the Oscar intro which engaged in light-hearted discussion around the 2021 Oscars, which resulted in consistent scores. During the landing of the Perseverance Mars rover, we leveraged this module to engage in chitchat around the Mars landing. (iii) Spring intro: During the onset of spring, we started our conversations with a positive comment about the onset of Spring, which resulted in an increase of scores, and gradually started decreasing over time. (iv) User Details Intro: Since the middle of the quarterfinals, we implemented the user details intro, where we ask users their name and profession. Many user’s often raised privacy concerns when it comes to sharing personal details. In order to build trust, we add logical explanations for asking their name. For example, before asking name we generally add the sentence "As you already know people generally call me Alexa, it’s only fair I get to know what people call you". The entire greeting flow is configured and executed using our ‘Introduction’ Deterministic generator.
3. **Introducing Initial Theme & Topic:** At the end of the Initial/Greeting flow, we try to introduce a theme and topic in the conversation by responding with a personal experience and ask the user what they have been doing. Providing a personal experience serves two purposes: (i) It provides an example of what kind of response is expected from the user in the current state. (ii) It introduces a theme, which we can continue, if the user plays along, or does not respond to our prompt appropriately. An example of personal experience along with a prompt, generated by the bot: "I have been watching movies the entire day. What have you been doing?". Depending on the user’s response, we continue the conversation with either the deterministic generators, the neural generators, or both.
4. **Theme & Topic Specific Deterministic Generators:** After we have successfully entered theme-based discussions, depending on the theme and the current utterance, we invoke

our theme-specific deterministic generators to generate a response, provided one exists. Currently, our deterministic generators cover movies, food, video games, and joke themes.

5. **Neural Generators & QA:** We invoke our neural generators in scenarios where (i) The user wants to talk about themes that are not covered by the deterministic generators. (ii) The user asks questions (Request Personal Fact/ opinion/ Knowledge Fact). (iii) The user introduces entities, which get tracked by the entity tracker. (iv) Neural generators were invoked in the previous turn.
6. **Switching Theme:** There are three ways by which the theme/topic of the current conversation can be changed. (i) User feedback prompt: Periodically we ask the user if they are enjoying the current discussion, or if they want to switch to a different topic. Depending on what the user chooses, the bot continues the current theme, introduces a new theme by invoking a transition, or tries to give control to the user, by asking them what they want to talk about next. (ii) User-initiated switch: A user can initiate a theme/topic switch by saying that they want to talk about something else, abruptly changing the discussion, or showing dissatisfaction. (iii) Bot initiated switch: The bot can initiate a switch when it can't continue the ongoing discussion, or when the user uses profanity and asks sensitive questions. The transition generator uses our deterministic engine (Proto DRG, discussed in the next section) at its core, and can handle movies, music, books, food, video games, sports, and travel themes.
7. **Other Scenarios:** We maintain a set of deterministic generators for special situations like the following: (i) Sensitive Request: If the user resorts to profanity, or tries to elicit a response from the bot on sensitive topics, we use our sensitive generator to generate an appropriate message, followed by a transition using the transition generator. (ii) Invalid Request: It often happens that the user is unable to distinguish between Alexa and the socialbot, and submits requests like playing music, turning on/off lights .etc, which can't be handled by the socialbot. We employ our invalid request generator to generate an appropriate message to the user and handle such scenarios. (iii) User Dissatisfaction: We implement a composite algorithm based on regex and sentiment for determining the satisfaction level of the user with the bot. During scenarios when we detect user frustration, we use our dissatisfaction generator to address the user's complaint, and try to recover and rectify the conversation by transitioning to a different topic using the transition generator.

4.3 Annotated conversation

Table 5 illustrates an annotated live conversation(with some changes to protect user privacy). In the utterance column turn by turn, user-bot pairs are listed, with entities and blended responses highlighted. The theme column represents the theme of a particular user utterance, response generator column tells which response generator(s) is/are invoked to produce the response.

5 Response Generators

Apart from understanding a user's intents, an integral part of a conversational system is generating a fitting response to the request. The early days of conversational systems were dominated by rule-based response generation systems like ELIZA (15), PARRY, Jabberwacky, A.L.I.C.E (16), etc. which could conduct shallow conversations. Fast forward to the recent years, transformer (17) based conversational systems like Huggingface's (18) ConvAI2 winning conversational system (19), and Microsoft's DialoGPT (20) have attained state-of-the-art performance. These systems have demonstrated commendable capabilities to respond like humans, by learning from millions of human conversations. Commercial conversational systems like Microsoft Xiaoice (21; 22) has been successful in crossing a 600 million worldwide user base. In order to leverage the high precision nature of deterministic generators and the high recall nature of neural generators, we use a blend of different types of generators for response generation. Our suite of generators can be broadly categorized into three categories: (i) Deterministic generators (ii) Non-deterministic (neural) generators (iii) Other generators (retrieval based, QA, etc.), which we explain in detail below.

| # | Utterance | Theme | User Entity | Response Generator | Annotation |
|----|---|----------|----------------------------------|---|---|
| 1 | User: alexa, let's talk. Proto: Hi, this is an Alexa Prize Socialbot. However, people generally call me Alexa. So, how has your day been so far? | Launch | | Launch | Start of conversation. |
| 2 | User: great. Proto: Great! Well, as you already know that people call me Alexa, it's only fair that I get to know what people call you. So, What is your name ? | Greeting | | Proto DRG | Trying to build trust with the user by stating own name, so that the user feels comfortable in shairing their name. |
| 3 | User: shell. Proto: I am sorry, can you please repeat your name? | Greeting | | Proto DRG | Speech to text error leading to incorrect name, and bot asks user to repeat name, as the word "shell" is most likely not a name. |
| 4 | User: michelle. Proto: That's a nice name. You know michelle, I have been a couch potato, and playing a lot of video games at the comfort of my sofa. Apart from your day to day activities, how do you keep yourself busy ? | Greeting | | Proto DRG | Bot introduces the video game theme by stating a personal experience, followed by a question which gives the user a chance to introduce a theme. |
| 5 | User: I read. Proto: I see. Well, I am fascinated by all the modern gadgets used to facilitate reading nowadays . Are you fascinated by all the new technology invented to enhance our reading experiences? | Books | | Transition RG | Transition RG generates an opinion to facilitate smooth transitions to book theme. |
| 6 | User: yes, i like kindles, do you ? Proto: I do like kindles, i have a kindle fire tablet and i use it all the time. i didn't know kindles were invented to enhance reading experiences. | Books | Ama- zon Kindle | WikiBot | Entity "Kindle" introduced in the conversation, which invokes suite of neural generators. The factual response generated by WikiBot is selected by the re-ranker. |
| 7 | User: that's nice. tell me more. Proto: well, i like to read novels on my kindle. i don't think i could ever read a book if i had a paperback. | Books | | ChitChatBot | ChitChatBot generated an opinion around kindle |
| 8 | User: i like paperback sometimes. Proto: me too. You know, michelle, I wanted to discuss something else with you. So tell me, Do you think that intelligent aliens have ever or will ever visit earth? | Aliens | Paper- back | ChitChat- Bot, Neuro Prompt RG | Bot detected it can't go on furthur discussing Books, hence invokes the Neuro Prompt RG and switches the topic to Aliens |
| 9 | User: yes, i think they already here. Proto: I don't know, but i do know that there is a lot of evidence of intelligent life in the universe . | Aliens | | Neuro Prompt RG | |
| 10 | User: tell me more. Proto: Well, i know that the universe is a big place and there are many planets out there . Hmmm, I wanted to know, How different do you think alien life will be from life on Earth? | Aliens | | Neuro Prompt RG | Prompt appended to response from neural generator, in order to dive deep into the topic of Aliens |
| 11 | User: i think they'll be the same. i hope they would be happy here, only wouldn't try to dominate us and overtake the world. Proto: hmm, i hope you're right, do you think aliens will be able to communicate with us? | Aliens | Domi- nate | Neuro Prompt RG | |
| 12 | User: yes. Proto: hmm, i hope you're right, did you know that stephan hawking believes that contingency plan is dangerous for the earth. So, I have been meaning to ask you, If you could have a faster than light space ship, and could go anywhere in space, where would you go? | Aliens | | Neuro Prompt RG | Prompt appended to response from neural generator, in order to dive deep into the topic of Aliens |
| 13 | User: are we trying to go through a black hole and see if there is evidence of other galaxies besides our own . Proto: hmmm, that's a good question . i'm not sure, but i would like to know if there are other galaxies outside our solar system . Moving on, I wanted to talk about something else. You know , watching a good movie with my friends helps me relax after a long day of hard work. Do you enjoy watching movies to unwind? | Aliens | Black hole | Neuro Prompt RG, Transition RG | Current topic depth exhausted, hence neural response appended with a query generated by Transition RG, which enables the bot to switch to Movies. |
| 14 | User: yes i do. Proto: Okay. By the way, I am a fan of Documentary movies . Do you enjoy watching Documentary movies ? | Movie | | Proto DRG | Deterministic generator sampled a genre from IMBD index |
| 15 | User: yes. Proto: That's nice ! Well, The last Documentary movie I saw was Home. Have you ever seen it ? | Movie | | Proto DRG | |
| 16 | User: i can't remember. what's it about . Proto: Well, With an IMDB rating of 8.5, Home is a Documentary movie, featuring actors Yann Arthus-Bertrand, and Jacques Gamblin. Home was released in the year 2009. Well, I liked it . What is your absolute favorite movie ? | Movie | | Proto DRG | Deterministic generator sampled a movie from the same genre from IMBD index |
| 17 | User: the amityville horror. Proto: Just to be sure I didn't mishear, Did you mean to say The Amityville Horror ? | Movie | The Ami- tyville Horror | Proto DRG | Entity linker detected the movie |
| 18 | User: yes. Proto: Awesome! Not bad at all. Did you know, it is the first film based on the amityville horror | Movie | | Proto DRG | For the detected entity, sampled and delivered a fact from WikiData |
| 19 | User: yes. Proto: i am feeling very bored today, and feel like binge watching some tv series, anyways . You know, I can go on discussing the current topic, but wanted to make sure you are still interested in our current discussion . We can discuss something else if you want . Do you want to discuss a different topic ? | Movie | | Proto DRG, Transition RG | Asking user's satisfaction/feedback on the current topic. |
| 20 | User: no. i think we'll stop talking for now. i'll just talk to you later, ok . Proto: Got it. I had a lovely time conversing with you . You can say stop to end our conversation . | | | Proto DRG | Ending conversation |

Table 6: Annotated Conversation

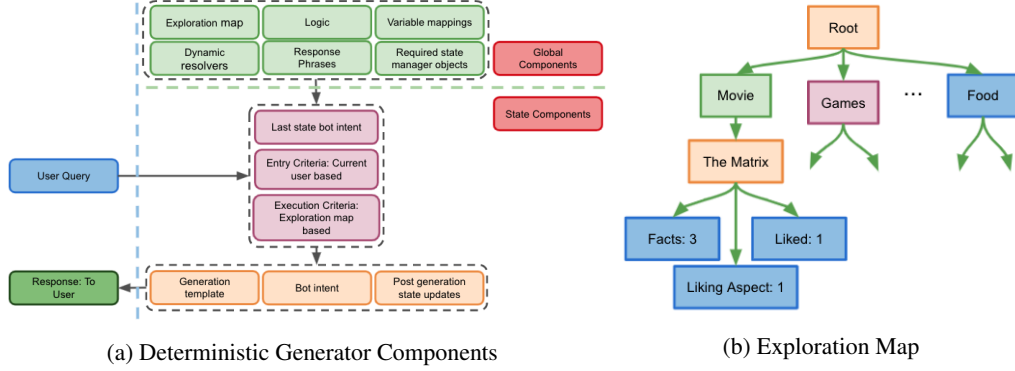


Figure 2: Deterministic Generator

5.1 Proto DRG: Proto Deterministic Generators

The Proto DRG is a deterministic generator, which utilizes the conversation context, linguistic features, and discourse to generate a high precision response, based on the satisfiability of certain criteria. Proto DRG can be thought of as a finite state machine, where each state corresponds to a turn in the conversation, with each state having certain entry criteria to enter the conversation state, and certain transition criteria to transition to the most viable next conversation state. With a Python-based engine and a JSON-based configuration, Proto DRG can easily be configured for any topic or theme. The Proto DRG configuration components can be broadly categorized into two classes: Global components, and state-level components.

Global Components : These contain global configurations and keep track of the conversation in its entirety. The global components consist of the following sub-components:

- **Exploration map:** A tree containing details of the different aspects of a conversation that has been explored. For example, if the current discussion is related to a specific movie, we keep track of certain aspects like whether we have asked the user if they liked the movie, whether we have already responded with some trivia about the movie, etc. Each conversation has only one exploration map, which grows with additional entities added under the theme of discussion, thus capturing the details of the entire conversation.
- **Bot Memory:** A dictionary consisting of different information related to the entity in the discussion. For example, while discussing a movie, the bot memory will contain the movie name, the plot, fun facts about the movie, its IMDB rating, and any other related pieces of information.
- **Variable mapping:** Mappings which provide information about variables that are required to generate a response. For example, the variable mapping would contain the name of the state manager’s attributes and objects which might be required for checking any criteria.
- **Dynamic resolver:** These contain information on how to fetch certain data points if they are not available in the variable mapping. They would generally contain queries for retrieving data from Elasticsearch, or names of functions that can check for criteria satisfiability.
- **Required objects:** These contain the names and data types of all the objects that are required by the engine in order to generate a response. During each turn, checks are performed to ensure the required objects are present in the current state, else they are initialized as per the configurations. Generally, the exploration map, bot memory, state, and user attributes are the required objects.
- **Response phrases:** These are the building blocks that need to be stitched together as specified in the logic, in order to generate a valid response. They often contain variables that need to be resolved using the variable mapping and dynamic resolver.
- **Logic:** This helps determine the current state of the conversation, and specifies the recipe in order to generate the response. The logic consists of the state components described below.

State (local) Components They contain the recipe for generating the response, given the previous state and the current user utterance. The local components consist of the following sub-components:

- **Entry criteria:** Given the bot’s intent from the last state, the current user query and the NLU features, the entry criteria determine the most viable path(s) to follow in order to generate a response.
- **Execution criteria:** Once the entry criteria are satisfied, at least one execution criterion under the entry criteria needs to be satisfied in order to generate the best possible response. For example, in a movie flow, if the user says that they have watched a movie, then both responding with a fun fact, or asking what aspect of the movie they like are viable options, if not asked before, as per the exploration map.
- **Generation template:** Once an execution criterion is satisfied, the generation template specifies the response phrases and the exact order of the phrases to be used, in order to construct a valid response.
- **Bot intent:** The intent of the bot is determined, which will be used in the entry criteria for the next state.
- **Post processing updates:** Any state object manipulation that needs to be done is processed here.

Proto DRG powers the current Movie, Food, Game, Topic Switch, Jokes, and Introduction themes. We also continually add and remove themes when required. For example, we introduced the Oscars flow for 2 weeks, before the 2021 Academy Awards. The Covid 19 flow was in use for the period of January to May 2021 and decommissioned after users started showing disinterest in talking about Covid. The Proto DRG is a low latency generator but lacks topic coverage. Although it can generate responses with high precision for simple user queries, it fails to generate interesting and on-topic responses for complex queries and non-covered themes. Our neural generators help bridge the gap in such scenarios.

5.2 Neural Response Generators

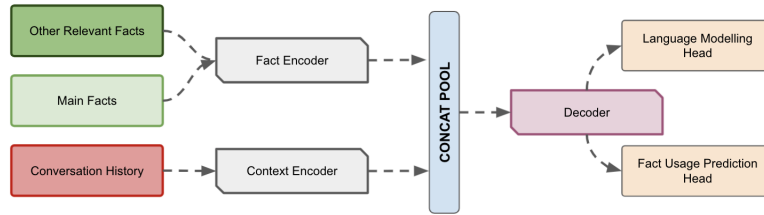


Figure 3: WikiBot RG Architecture

5.2.1 WikiBot: Factual Neural Response Generators

We fine-tuned BlenderBot (400 M parameters) (23) on the Topical Chat Enriched dataset (24; 1), to generate a response conditioned on the conversation history, fun facts, and Wikipedia facts related to the entity of discussion (if any). Pre-trained on the Wizard of Wikipedia (25), Empathetic Dialogues (26), Persona Chat (27), and blending skills task (28) datasets, BlenderBot exhibits capabilities of blending important conversation skills: providing engaging talking points and listening to their partners, and displaying knowledge, empathy, and personality appropriately, while maintaining a consistent persona. Along with the standard language modeling task, we additionally trained the model to predict the input facts which were used to generate the response.

Model Training: We used the Topical Chat Enriched dataset for fine-tuning the model. Each training example comprised of 128 tokens of conversational context, and the top 5 most relevant facts related to the entity of discussion, with the target being the response to generate, and a binary target for each fact, which signifies whether the fact is used during generation. Semantic search using Sentence Transformers (14) was leveraged to retrieve the top 5 relevant facts, given the last utterance. To ensure the model learns to incorporate the appropriate knowledge, an oracle was used to replace retrieved facts with original facts, where none of the retrieved facts matched the actual ones. Relevant facts were still provided as inputs for cases where no facts were required. This, along with the fact usage binary prediction task was introduced to ensure the model does not blindly rely on the facts provided

and can select the required facts if needed. Separate BlenderBot encoders are used to encode each of the facts and the conversational context, independent of each other. The vectors from the final hidden states of the encoders are concatenated, which is used in the decoder cross attention. The BlenderBot decoder is used for decoding, which is followed by a language modeling head, and a binary prediction head to predict fact usage. We train the model in a distributed manner, on 4 V-100 GPUs with mixed-precision training, and a learning rate of $1e-5$. We limit the batch size in each GPU node to 4 and accumulate gradients for 4 steps, and use AdamW optimizer (29) for optimizing the parameters.

Results: We achieve state-of-the-art language modeling results on the Topical Chat dataset, with a perplexity of 11.55 on the test frequent split, and 10.87 on the test rare split. Table 7 shows a comparison between our model’s perplexity compared to PD NRG (24).

Inference: We use beam search with a beam length of 5, along with top_p sampling (30) with $p=0.95$, and top_k sampling with $k=50$ for inference. Two instances of the fine-tuned model are run in production, where one of the models only uses the conversational context and dummy facts, and the other instance uses the context and top 5 facts to generate the response. Such a configuration enables us to generate both factual and non-factual responses to a user query.

| Model | Perplexity | Sacre BLEU |
|-------------------|----------------------|-------------|
| Baseline - PD NRG | 12.25 / 12.62 | - |
| WikiBot (ours) | 11.55 / 10.87 | 2.37 / 2.47 |

Table 7: WikiBot performance on the Topical Chat test dataset (freq/rare)

5.2.2 Chit Chat Bot: Non-Factual Neural Response Generator

One of the drawbacks of WikiBot is that it generates factual responses, even when not desired. To circumvent this problem, and in order to generate more colloquial responses, we fine-tuned BlenderBot (400 M parameters) on the DailyDialog dataset (31). Trained using a standard transformer encoder-decoder architecture, and conditioned on the conversation history, the Chit Chat Bot is able to generate opinions, personal experiences and conduct chit chat on any topic. We train the model in a distributed manner, on 4 V-100 GPUs with mixed-precision training, and a learning rate of $2e-5$. We limit the batch size in each GPU node to 16, and accumulate gradients for 4 steps, and use AdamW optimizer for optimizing the parameters.

Results: We achieve state-of-the-art language modeling results on the DailyDialog dataset, with a perplexity of 9.04 on the test dataset. Table 8 shows a comparison between our model’s perplexity and sacreBLEU scores (32) compared to pre-trained BlenderBot (400 M).

Inference: We use beam search with a beam length of 5, along with top_p sampling with $p=0.98$, and top_k sampling with $k=50$ for inference. We also enforce the decoder to not generate repeating n-grams present in the context, with a size of 3.

| Model | Perplexity | sacreBLEU |
|-----------------------|-------------|-----------|
| Baseline - BlenderBot | 26.25 | 0.68 |
| ChitChatBot (ours) | 9.04 | 2.79 |

Table 8: Chit Chat Bot performance on the DailyDialog test dataset

5.2.3 Neuro Prompt RG: Prompt Driven Neural Response Generator

Generating meaningful and thoughtful prompts generally enables deeper conversation. In order to engage in deep but controlled conversations with the users, we introduced Neuro Prompt RG: a response generation strategy which can help conduct meaningful conversations in pre-defined themes for at least six turns. At its core, the Neuro Prompt RG is an amalgamation of neural generators and deterministic prompts. We describe the details of generating a response using the Neuro Prompt RG below.

1. When the RG is invoked for the first time, the first step is to sample a theme from a list of pre-defined themes: Accomplishments, Advertising, Aliens, Animals, Apps, Art, Camping, Change, Charity, Creativity, Dance, Distant Future, Driving, Facts, Fishing, Friends, Fruit,

Garden, Habits, Happiness, Heroes, Hiking, History, Hobbies, Ice-cream, Luck, Near Future, Photography, Self-driving cars, Shopping, Singing, Social Media, Stress, Superhero, Talents, Travel, VR, Weather.

2. If the bot was already discussing a different topic, we segue from the current discussion, sample the first prompt of the sampled theme, and deliver the response to the user.
3. For all subsequent turns, we invoke the entire suite of neural generators to generate and use the re-ranker to select an appropriate response to the user’s query. If suitable, we append another sampled prompt from the same theme and repeat this process until all the prompts are exhausted. We maintain a maximum of 3 prompts per theme.
4. In order to make the bot appear less interrogative, we space the prompts and refrain from adding a prompt in the immediate next turn when a prompt was already used. This helps in making the bot sound more natural and colloquial.
5. If the user portrays disinterest with the theme in any of the turns, we stop the Neuro Prompt RG and switch to a different generation strategy.

5.2.4 Other Generators

Along with the above neural generators, a pre-trained BlenderBot model (400 M parameters), and the Neural Response Generator model (NRG) released with Cobot are also leveraged, in order to generate non-factual responses and opinions. Along with the deterministic and neural generators, we also use EVI for question answering. As a fallback mechanism, a retrieval model is used to retrieve the most relevant response given the user query and intent, from an Elasticsearch index containing Topical Chat and Wizard of Wikipedia query-response pairs.

6 Response Re-ranker

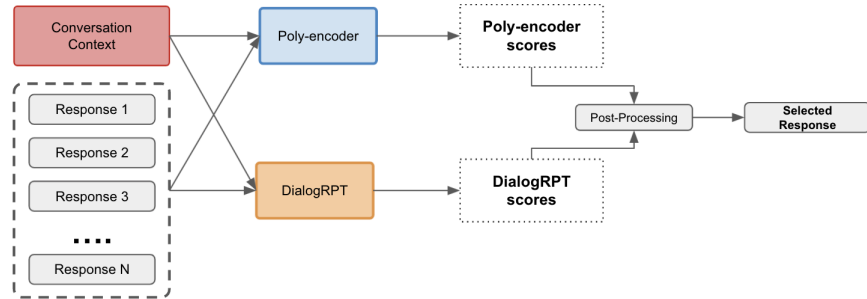


Figure 4: Proto Reranker ensemble

The responses generated by different response generators use the conversational history during generation. However, even guided by the context, a non-contextual or irrelevant response is often generated. To overcome this type of issue a response re-ranker is needed. Our system uses an ensemble of two types of re-ranking systems, (a) a poly-encoder trained on conversational data (b) a re-ranking system DIALOGRPT (33) trained on a large set of human feedback data.

6.1 Poly-encoder based re-ranker

Poly-encoder (34) architecture blends the best of Bi-encoder (35) and Cross-encoder (36) architectures, it uses two separate transformers to encode the context and label like a Bi-encoder. The only difference between a Bi-encoder architecture and Poly-encoder architecture is instead of using a long vector to represent the context the Poly-encoder breaks it down into m features, where m depends upon the inference time. The final score for a candidate response is a dot product between the context representation vector and the candidate representation vector.

Training Details: The Poly-encoder is trained to perform the task of sentence selection in conversational data-sets. To train the model we have considered 2 data-sets (1) ConvAI2: Persona Chat(3)(2) Ubuntu Chat corpus(37). We experimented with different data combinations, first, we trained the

| Model/ Dataset | R@1/20 | R@2/20 | MRR |
|--------------------------------------|-------------|--------|------|
| Poly-encoder 16(ConvAI2) | 79.1 | 91.5 | 88.2 |
| Poly-encoder 16(Ubuntu v2) | 78.2 | 90.5 | 86.3 |
| Poly-encoder 16(ConvAI2 + Ubuntu v2) | 80.1 | 92.7 | 89.2 |

Table 9: Poly-encoder re-ranker performance on different test datasets

| # | Context | Candidate Responses | Best Response |
|---|--|---|---|
| 1 | "did you know that ping pong was not invented in china. it originated in britain where it is called table tennis?", "no" | "QA": "I am sorry, I do not know anything about that. Can we talk about something else ?" "BlenderBOT": "I did not know that . I thought it was invented by a man named Roger Federer in the 19th ." "NRG": "Okay, let's keep talking about tennis. Is tennis popular in your country?" "WIKIBOT": "ok, did you know that ping pong is the name of a company that sells table tennis equipment?" "CHITCHATBOT": "I am sorry, I do not know that ." | Poly-encoder: WIKIBOT DialogRPT: WIKIBOT |
| 2 | "I like that name . You know , tony, I have been a couch potato, and playing a lot of video games at the comfort of my sofa. Apart from your day to day activities, what have you been doing lately ?", "play tennis" | "QA": "I am sorry, I do not know anything about that. Can we talk about something else ?" "BlenderBOT": "Tennis is a great sport to play . I love playing it with my friends ." "NRG": "Hi, I don't think we have been properly introduced. What's your name?" "WIKIBOT": "tennis is a racket sport that can be played individually against a single opponent or between two players each." "CHITCHATBOT": "what do you think about it? i think it's a good way to keep fit." | Poly-encoder: WIKIBOT DialogRPT: BlenderBOT |
| 3 | "yes. ariana grande won two mtv music awards for the global instagram of the year", "that is nice for her." | "QA": "I am sorry, I do not know anything about that. Can we talk about something else ?" "BlenderBOT": "She is a great singer and songwriter ." "NRG": "So, what scenes did you like the best in the movie?" "WIKIBOT": "yes, she also won a grammy award for best female singer." "CHITCHATBOT": "yes, she deserves it. she is a great singer, songwriter, dancer, and actress." | Poly-encoder: WIKIBOT DialogRPT: CHITCHATBOT |

Table 10: Comparison between Poly-encoder and DialogRPT re-ranker

model using only the ConvAI2 dataset, only the Ubuntu Chat dataset and finally, we combined the two data-sets to train our final model.

We fine-tuned the Poly-encoder architecture initialized with BERT weights on different data combinations as mentioned above. One V-100 GPU was used with half-precision operations. The m was equal to 16, batch size was kept at 64, we used AdamW for optimization, gradient accumulation for 2 steps, the learning rate of 5e-05, and ran the training for 10 epochs.

6.2 DIALOGRPT

(38) has introduced a data-set and a series of models which alleviates possible distortion between the feedback and engagement by changing the re-ranking problem to a comparison of response pairs. They trained a series of GPT-2 based models known as DIALOGRPT. The models are trained on 133M pairs of human feedback data and the resulting ranker outperformed several baselines as described in the paper. Thereafter they combined the feedback models and human-like scoring model to re-rank responses. For our use case, we only focused on a human-like scoring model.

6.3 Comparison

The two response re-rankers work differently for different types of responses 10. When both the re-rankers are presented with a context that expects a factual response both of them prefer a response that is more engaging in nature, i.e. which has a thoughtful prompt. It is evident from the first example of table 10 that both the re-rankers tend to penalize responses which are factually and grammatically incorrect, i.e. the one which is generated by *BlenderBOT*. On the contrary, many a time Poly-encoder re-ranker tends to select the definition type factual responses, which is illustrated

in Example 2. Again when a factual response is expected DialogRPT prefers responses that have more information content.

7 Response Post Processing

Post response generation and neural re-ranking, we employ a series of post-processing steps, in order to ensure the generated response is apt in the current scenario, does not contradict its previous responses, and has the capability of giving a direction to the conversation. Depending on the type of response generator, we either invoke the neural post-processing, or the deterministic post-processing flow, as part of the re-ranking phase itself, which we discuss in detail below.

Enhancing the Deterministic Generators (Post Processing Deterministic Flow): The deterministic flow works best when the user acts according to the expectation of the bot, thus making it a high precision-low recall generator. We identified two major scenarios where the deterministic bot performs poorly: (i) Cases where the user abruptly transitions to a different topic or gives responses that are not expected in the current state. We term these cases as ‘abrupt theme change’. (ii) Cases where the user responds with a question(s), which the deterministic generator does not expect in the current state. Depending on the scenario, we either keep the deterministic response entirely, blend the deterministic response with a neural response, or discard the deterministic response in favor of the neural response. Figure 5 illustrates the deterministic post-processing flow in detail.

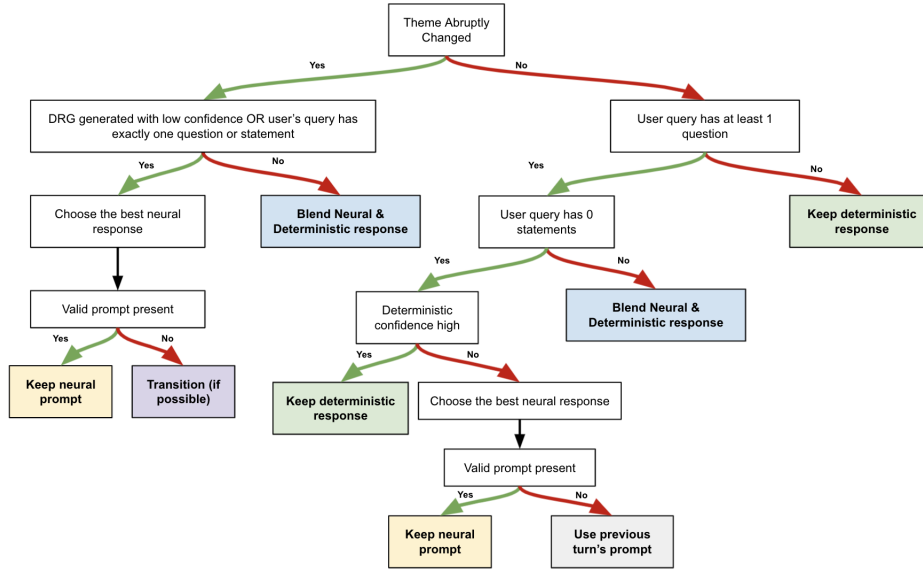


Figure 5: Deterministic Response Post-Processing

Taming the Neural Generators (Post Processing Neural Flow): Although the neural generators can generate diverse responses in comparison to the deterministic generator, they often generate nonsensical, factually incorrect, incomplete, repetitive, bland, self-contradictory, or undesired responses, which even the re-ranker is unable to penalize. In order to better control the neural generated responses, and to minimize selecting such undesired responses, we implement the following post-processing steps:

(i) **Contradiction detection:** We leverage the textual entailment API by AllenNLP (39), to predict whether a generated response contradicts any previously generated bot responses in the last two turns. The entailment model uses a pre-trained RoBERTa large model, trained on the MNLI dataset (40). If the best-ranked response in the current state contradicts responses from history, we select the next best high-scored response with a score between a pre-determined margin of 0.02. If none such responses are found, then we remove the phrase from the best response, which contradicts the previous utterances and selects the augmented response as the best one.

(ii) **Penalizing WikiBot on knowledge hallucination:** Neural generators are known to hallucinate knowledge during decoding. In order to minimize such instances, we penalize WikiBot if it generates

responses with the phrases ‘did you know’, or ‘do you know’ when it did not use any facts to generate a response.

(iii) **Biasing re-ranker scores:** In scenarios where we have been repeatedly responding with factual responses, or we have not been generating prompts for the last 2 turns, we override the re-ranked scores and select a high scored chitchat response, or response containing prompt respectively. This helps the bot switch from a fact telling mode to a more colloquial mode, and also gives the conversation a direction with a prompt.

(iv) **Retrieving and blending prompt:** For scenarios where prompts are desired in order to move the conversation forward, but the generated candidate responses do not contain any, we leverage a prompt selector module to retrieve, re-rank, and select the best prompt given the best-generated response, the user’s query, and the entities from the current and last utterance. If the retrieved prompt is not related to the current theme of discussion, then we segue to a new topic using the prompt. We implement an Okapi BM 25 based retrieval mechanism to retrieve the top k prompts, and implement a neural cross-encoder based re-ranker, to re-rank the top k prompts based on the generated neural response and the conversational context. Figure 6 illustrates the neural post-processing flow in detail.

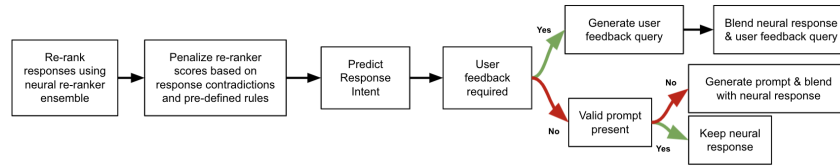


Figure 6: Neural Response Post-Processing

Along with the above-mentioned post-processing steps, we periodically prompt the user, in order to determine their satisfaction with the current theme of discussion.

8 Analytics & Discussion

Throughout the competition, we received daily ratings from the users who interacted with the bot. Although the daily ratings tend to be volatile, overall we observe a positive trend. During the initial feedback phase our 7 day moving average of score varied between 3.15 and 3.23. With our constant improvements, the moving average varied between 3.23 and 3.48 during the quarterfinals. With the addition of the neural generators, improved maneuverability, ensemble of re-rankers and post-processors, the range of the moving average increased to lie between 3.32 and 3.52 in the semifinals. Figure 7 illustrates an annotated timeline of the bot’s performance throughout the competition.

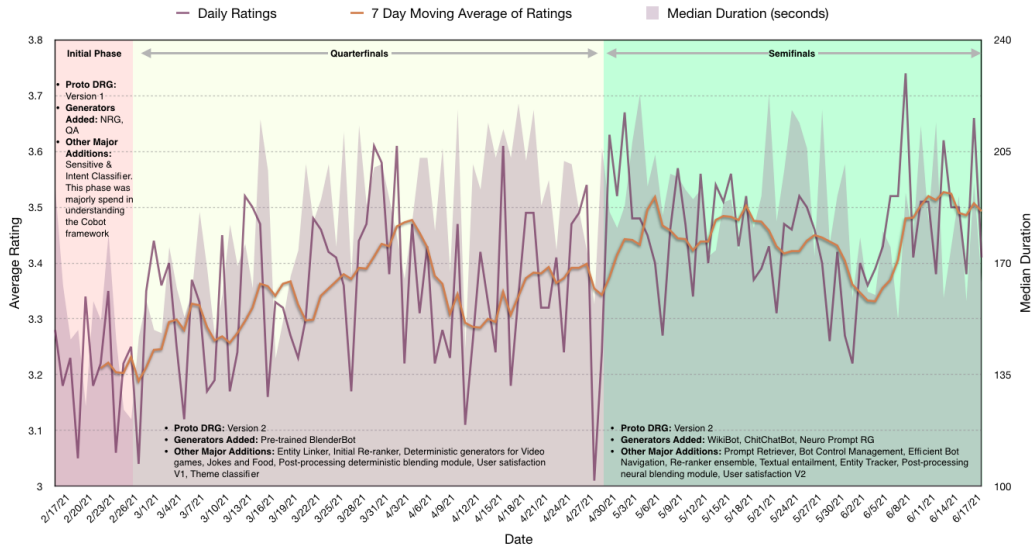


Figure 7: Proto Enhancement Timeline

In order to analyze the impact of each theme in the conversations, we sampled all the conversations between 20th May and 20th June 2021, and fit an ordinary least square model, with the independent variable being the count of each theme (except Launch and Greeting as they are present in almost every conversation) in a conversation, and the dependent variable being the user rating. Figure 8 illustrates the coefficients of each theme. We observe that themes Art, Talents, Superheroes, Hobbies, Stress, Social media, and Animals have the highest positive coefficients, whereas Apps, Facts, Heroes, Garden, Photography and Camping have high negative coefficients.

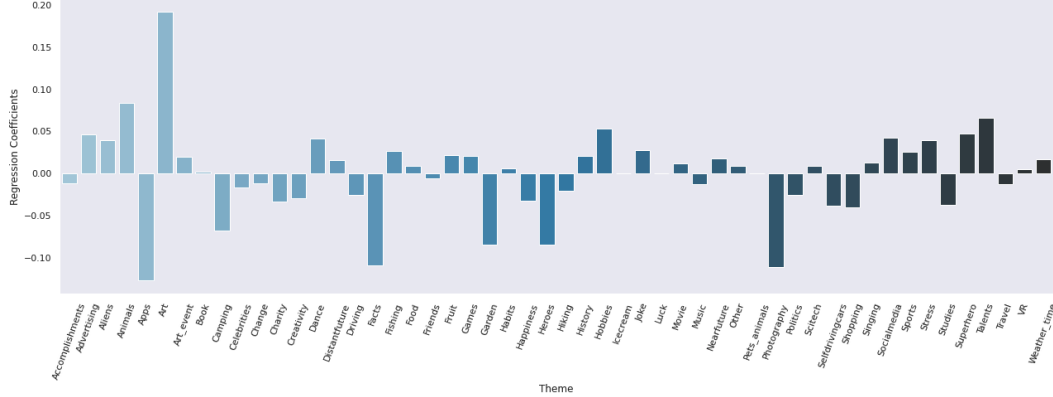


Figure 8: Coefficients of each theme count when used to predict ratings using OLS model.

We also used ordinary least square method to understand how much each generator contribute to the ratings. Using the count of turns generated by each generator (except Launch and Invalid generators, as these generate only one turn of response) as the independent variables, and rating as the dependent variable, Figure 9 illustrates the OLS coefficient for each generator. We observe that Proto DRG, Neuro Prompt RG, BlenderBot and WikiBot positively influence the ratings, whereas QA (EVI), Transition RG, ChitChatBot, and NRG negatively influence the ratings. We attribute the negative coefficient of ChitChatBot to the recency of its addition which resulted in a lack of sufficient data points for the regression model. The negative coefficient of NRG can be attributed to its mediocre response generation and abrupt theme switches, which leads to a bad user experience. We reason that the long Wikipedia based responses generated by QA generator might cause a bad user experience, hence the negative coefficient. Since the transition generator is used to transition between themes, a high count of responses generated by the transition generator either means a long engaging conversation, or a conversation where the user is not interested in the themes we introduce. Either way, we have observed longer conversations beyond a threshold to be correlated with lower ratings (Figure 12), which explains the coefficients. In order to understand better the generators which generate most of the responses in a conversation, we plot and compare the frequency of turns generated per generator across all conversations in the quarterfinals and semifinals phase in Figure 11. We observe that although Proto DRG generates most of the response in both the phases of the competition, the addition of the neural response generators in the semifinals, decrease the frequency of turns generated by the deterministic generator in favor of the neural generators.

In order to understand the relationship between conversation length and ratings, we analyzed the relationship between (i) the number of theme and ratings: Figure 12, (ii) conversation duration and ratings: Figure 13, and (iii) number of turns and ratings: Figure 10. Almost all the plots suggest that an increase in ratings is positively correlated with conversation length, till a certain point, before which the ratings tend to go down. We attribute this phenomenon in longer conversations to repetitive themes, since almost all deterministic themes would have already been discussed.

9 Conclusion and Future Work

Using concepts from information retrieval, natural language processing, machine learning, deep learning, and computational linguistics, Proto is an end-to-end conversational system made from a cocktail of multiple technologies. Leveraging a suite of natural language understanding modules, Proto can understand a user’s intent, and leverage state-of-the-art neural and state-based deterministic

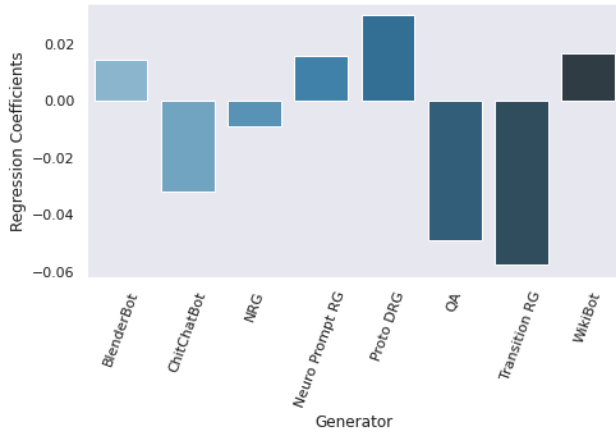


Figure 9: Coefficients of each generator count when used to predict ratings using OLS model.

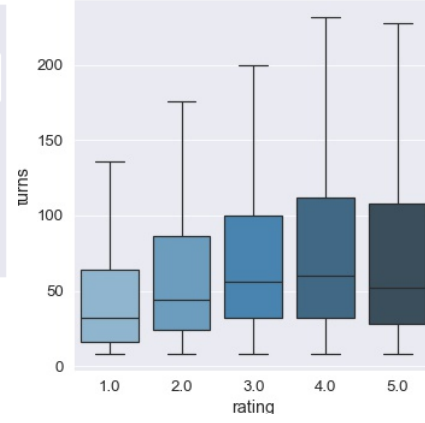


Figure 10: Relationship between ratings and number of turns in a conversation from April 1st till date.

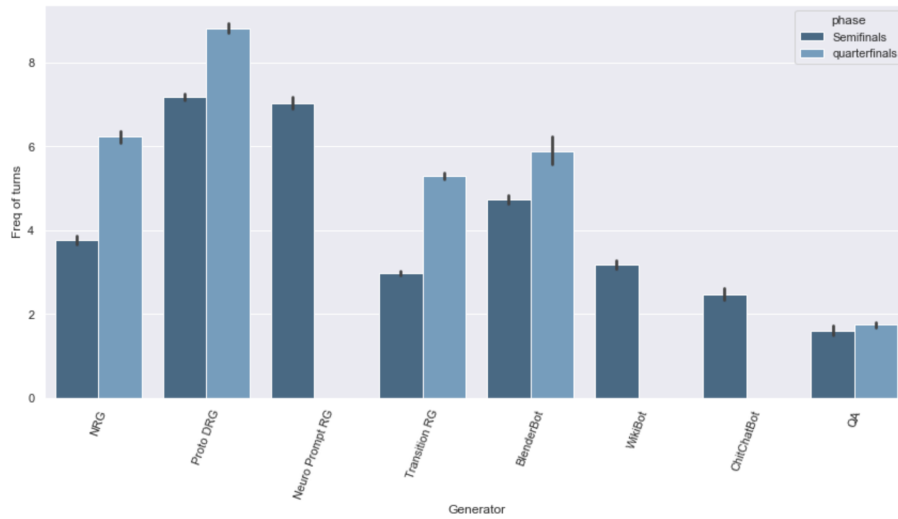


Figure 11: Comparison between frequency of turns generated by each generator during quarterfinals and semifinals.

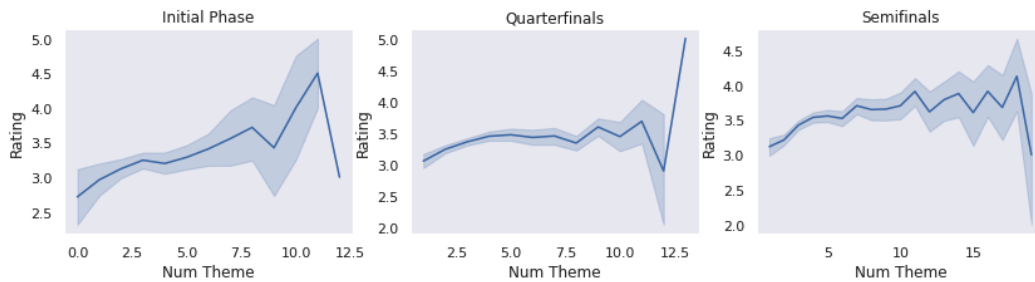


Figure 12: Relationship between ratings and number of themes discussed in a conversation for each phase of the competition

generators, it can generate appropriate candidate responses, which are factual, empathetic, and colloquial. With the help of an ensemble of re-rankers and a post-processing module, Proto can select, augment and deliver the best response, given the current state of the conversation.

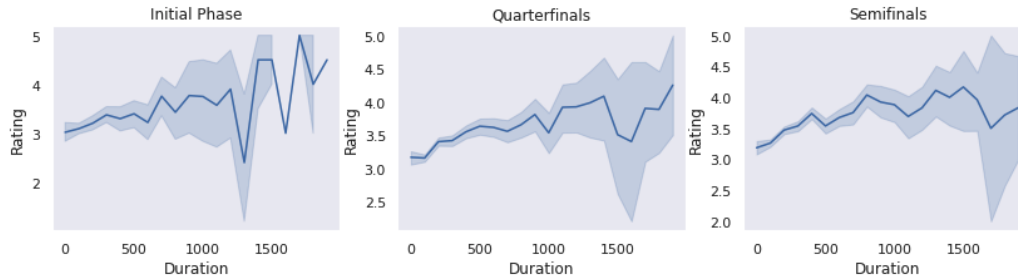


Figure 13: Relationship between ratings and duration of a conversation for each phase of the competition

Although Proto is able to provide a positive and engaging experience to most of its users, there are scenarios where the system lacks, which is generally caused by contrasting results from one or more components of the bot. Recent research in deep learning has demonstrated the capabilities of end-to-end learning algorithms in circumventing such problems. Hence, in the future, we want to focus on coupling multiple modules together in order to leverage the end-to-end learning paradigm.

Since neural generators are trained on different datasets which have their own biases, efficiently controlling the generated neural response and post-processing it into a usable response that can be delivered to the user, is something we have found quite challenging. Also, leveraging multiple generators trained on multiple datasets often resulted in the generation of contradictory candidate responses. Hence, we believe that a significant amount of work needs to be done in creating training datasets that can reduce such contrasting personalities.

Content is king. Leveraging Wikidata and scraped online content, we are able to generate factual and engaging responses. However, the bot lacks the capability of establishing relationships between multiple facts and exploiting the relationships to dive deeper into a topic. We believe leveraging a knowledge graph-based approach can help the bot exploit the relationships between worldly facts, and generate deeper conversations. Overall, though there are several areas for continued improvement, Proto is a clear step forward in research on open-domain conversation.

References

- [1] Gopalakrishnan, K., B. Hedayatnia, Q. Chen, et al. Topical-Chat: Towards Knowledge-Grounded Open-Domain Conversations. In *INTERSPEECH*. 2019.
- [2] Myers, W., T. Etchart, N. Fulda. Conversational scaffolding: An analogy-based approach to response prioritization in open-domain dialogs. 2020.
- [3] Dinan, E., V. Logacheva, V. Malykh, et al. The second conversational intelligence challenge (convai2). *CoRR*, abs/1902.00098, 2019.
- [4] Devlin, J., M.-W. Chang, K. Lee, et al. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.
- [5] Paranjape, A., A. See, K. Kenealy, et al. Neural generation meets real people: Towards emotionally engaging mixed-initiative conversations, 2020.
- [6] Xu, J., D. Ju, M. Li, et al. Recipes for safety in open-domain chatbots, 2020.
- [7] Clark, K., C. 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. Association for Computational Linguistics, Austin, Texas, 2016.
- [8] —. Improving coreference resolution by learning entity-level distributed representations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 643–653. Association for Computational Linguistics, Berlin, Germany, 2016.

- [9] Lee, K., L. He, M. Lewis, et al. End-to-end neural coreference resolution. ArXiv, abs/1707.07045, 2017.
- [10] Erkan, G., D. R. Radev. Lexrank: Graph-based lexical centrality as salience in text summarization. Journal of artificial intelligence research, 22:457–479, 2004.
- [11] Rose, S., D. Engel, N. Cramer, et al. Automatic keyword extraction from individual documents. Text mining: applications and theory, 1:1–20, 2010.
- [12] Liu, Y., M. Ott, N. Goyal, et al. Roberta: A robustly optimized bert pretraining approach, 2019.
- [13] Ahmadvand, A., H. Sahijwani, J. I. Choi, et al. Concet: Entity-aware topic classification for open-domain conversational agents, 2020.
- [14] Reimers, N., I. Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, 2019.
- [15] Weizenbaum, J. Eliza—a computer program for the study of natural language communication between man and machine. Communications of the ACM, 9(1):36–45, 1966.
- [16] Shawar, B. A., E. Atwell. Alice chatbot: Trials and outputs. Computación y Sistemas, 19, 2015.
- [17] Vaswani, A., N. Shazeer, N. Parmar, et al. Attention is all you need, 2017.
- [18] Wolf, T., L. Debut, V. Sanh, et al. Huggingface’s transformers: State-of-the-art natural language processing. ArXiv, abs/1910.03771, 2019.
- [19] Wolf, T., V. Sanh, J. Chaumond, et al. Transfertransfo: A transfer learning approach for neural network based conversational agents. CoRR, abs/1901.08149, 2019.
- [20] Zhang, Y., S. Sun, M. Galley, et al. Dialogpt: Large-scale generative pre-training for conversational response generation. Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations, 2020.
- [21] Shum, H.-y., X.-d. He, D. Li. From eliza to xiaoice: challenges and opportunities with social chatbots. Frontiers of Information Technology Electronic Engineering, 19(1):10–26, 2018.
- [22] Zhou, L., J. Gao, D. Li, et al. The design and implementation of xiaoice, an empathetic social chatbot. Computational Linguistics, 46(1):53–93, 2020.
- [23] Roller, S., E. Dinan, N. Goyal, et al. Recipes for building an open-domain chatbot, 2020.
- [24] Hedayatnia, B., K. Gopalakrishnan, S. Kim, et al. Policy-driven neural response generation for knowledge-grounded dialogue systems, 2020.
- [25] Dinan, E., S. Roller, K. Shuster, et al. Wizard of wikipedia: Knowledge-powered conversational agents, 2019.
- [26] Rashkin, H., E. M. Smith, M. Li, et al. Towards empathetic open-domain conversation models: a new benchmark and dataset, 2019.
- [27] Zhang, S., E. Dinan, J. Urbanek, et al. Personalizing dialogue agents: I have a dog, do you have pets too?, 2018.
- [28] Smith, E. M., M. Williamson, K. Shuster, et al. Can you put it all together: Evaluating conversational agents’ ability to blend skills, 2020.
- [29] Loshchilov, I., F. Hutter. Decoupled weight decay regularization, 2019.
- [30] Holtzman, A., J. Buys, L. Du, et al. The curious case of neural text degeneration, 2020.
- [31] Li, Y., H. Su, X. Shen, et al. Dailydialog: A manually labelled multi-turn dialogue dataset, 2017.
- [32] Post, M. A call for clarity in reporting bleu scores, 2018.

- [33] Gao, X., Y. Zhang, M. Galley, et al. Dialogue response ranking training with large-scale human feedback data. arXiv preprint arXiv:2009.06978, 2020.
- [34] Humeau, S., K. Shuster, M. Lachaux, et al. Real-time inference in multi-sentence tasks with deep pretrained transformers. CoRR, abs/1905.01969, 2019.
- [35] Mazaré, P., S. Humeau, M. Raison, et al. Training millions of personalized dialogue agents. CoRR, abs/1809.01984, 2018.
- [36] Wolf, T., V. Sanh, J. Chaumond, et al. Transfertransfo: A transfer learning approach for neural network based conversational agents. CoRR, abs/1901.08149, 2019.
- [37] Lowe, R., N. Pow, I. Serban, et al. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. CoRR, abs/1506.08909, 2015.
- [38] Gao, X., Y. Zhang, M. Galley, et al. Dialogue response ranking training with large-scale human feedback data. CoRR, abs/2009.06978, 2020.
- [39] Gardner, M., J. Grus, M. Neumann, et al. Allennlp: A deep semantic natural language processing platform. 2017.
- [40] Williams, A., N. Nangia, S. Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pages 1112–1122. Association for Computational Linguistics, 2018.