A Deep Reinforcement Learning Chatbot

Iulian V. Serban, **Chinnadhurai Sankar**, **Mathieu Germain**, **Saizheng Zhang**, **Zhouhan Lin**,
Sandeep Subramanian, **Taesup Kim**, **Michael Pieper**, **Sarath Chandar**, **Nan Rosemary Ke**,
Sai Mudumba, **Alexandre de Brebisson Jose M. R. Sotelo**, **Dendi Suhubdy**,
Vincent Michalski, **Alexandre Nguyen**, **Joelle Pineau** and **Yoshua Bengio**
Montreal Institute for Learning Algorithms, Montreal, Quebec, Canada

Abstract

We present MILABOT: a deep reinforcement learning chatbot developed by the Montreal Institute for Learning Algorithms (MILA) for the Amazon Alexa Prize competition. MILABOT is capable of conversing with humans on popular small talk topics through both speech and text. The system consists of an ensemble of natural language generation and retrieval models, including template-based models, bag-of-words models, sequence-to-sequence neural network and latent variable neural network models. By applying reinforcement learning to crowdsourced data and real-world user interactions, the system has been trained to select an appropriate response from the models in its ensemble. The system has been evaluated through A/B testing with real-world users, where it performed significantly better than competing systems. Due to its machine learning architecture, the system is likely to improve with additional data.

1 Introduction

Dialogue systems and conversational agents - including chatbots, personal assistants and voice-control interfaces - are becoming ubiquitous in modern society. Examples of these include personal assistants on mobile devices, technical support help over telephone lines, as well as online bots selling anything from fashion clothes and cosmetics to legal advice and self-help therapy. However, building intelligent conversational agents remains a major unsolved problem in artificial intelligence research.

In 2016, Amazon.com Inc proposed an international university competition with the goal of building a socialbot: a spoken conversational agent capable of conversing coherently and engagingly with humans on popular topics, such as entertainment, fashion, politics, sports, and technology. The socialbot converses through natural language speech through Amazon's Echo device (Stone & Soper 2014). This article describes the models, experiments and final system (MILABOT) developed by our team at University of Montreal.[1] Our main motivation for participating has been to help advance artificial intelligence research. To this end, the competition has provided a special opportunity for training and testing state-of-the-art machine learning algorithms with real users (also known as *machine learning in the wild*) in a relatively unconstrained setting. The ability to experiment with real users is unique in the artificial intelligence community, where the vast majority of work consists of experiments on fixed datasets (e.g. labeled datasets) and software simulations (e.g. game engines). In addition, the computational resources, technical support and financial support provided by Amazon has helped scale up our system and test the limits of state-of-the-art machine learning methods. Among other things, this support has enabled us to crowdsource $200,000$ labels on Amazon Mechanical Turk and to maintain over 32 dedicated Tesla K80 GPUs for running our live system.

[1] Our team is called MILA Team, where MILA stands for the Montreal Institute for Learning Algorithms.

1st Proceedings of Alexa Prize (Alexa Prize 2017).

Our socialbot is based on a large-scale ensemble system leveraging deep learning and reinforcement learning. We develop a new set of deep learning models for natural language retrieval and generation — including recurrent neural networks, sequence-to-sequence models and latent variable models — and evaluate them in the context of the competition. These models are combined into an ensemble, which generates a candidate set of dialogue responses. Further, we apply reinforcement learning — including value function and policy gradient methods — to train the system to select an appropriate response from the models in its ensemble. In particular, we propose a novel reinforcement learning procedure, based on estimating a Markov decision process. Training is carried out on crowdsourced data and on interactions recorded between real-world users and a preliminary version of the system. The trained systems yield substantial improvements in A/B testing experiments with real-world users.

In the competition semi-finals, our best performing system reached an average user score of 3.15 on a scale $1 - 5$, with a minimal number of hand-crafted states and rules and without engaging in *non-conversational activities* (such as playing games or taking quizzes).[2] The performance of this best system is comparable to some of the top systems in the semi-finals.[3] Further, the same system averaged a high $14.5 - 16.0$ turns per dialogue. This improvement in back-and-forth exchanges between the user and system suggests that our system is likely to be the most interactive and engaging system among all systems in the competition. Finally, the system is bound to improve with additional data, as nearly all system components are learnable.

2 System Overview

Early work on dialogue systems (Weizenbaum 1966, Colby 1981, Aust et al. 1995, McGlashan et al. 1992, Simpson & Eraser 1993) were based mainly on states and rules hand-crafted by human experts. Modern dialogue systems typically follow a hybrid architecture, combining hand-crafted states and rules with statistical machine learning algorithms (Suendermann-Oeft et al. 2015, Jurčíček et al. 2014, Bohus et al. 2007, Williams 2011). Due to the complexity of human language, however, it will probably never be possible to enumerate states and rules required for building a socialbot capable of conversing with humans on open-domain, popular topics. In contrast to such rule-based systems, our core approach is built entirely on statistical machine learning. We believe that this is the most plausible path to artificially intelligent conversational agents. The system architecture we propose aims to make as few assumptions as possible about the process of understanding and generating natural human language. As such, the system utilizes only a small number of hand-crafted states and rules. However, every system component has been designed to be optimized (trained) using machine learning algorithms. These system components will be trained first independently on massive datasets and then jointly on real-world user interactions. This way, the system will learn all relevant states and rules for conducting open-domain conversations implicitly. Given an adequate amount of examples, such a system should outperform systems based on hand-crafted states and rules. Further, the system will continue to improve in perpetuity with additional data.

Our system architecture is inspired by the success of ensemble-based machine learning systems. These systems consist of many independent sub-models combined intelligently together. Examples of such ensemble systems include the winner of the Netflix Prize (Koren et al. 2009), utilizing hundreds of machine learning models to predict user movie preferences, and IBM Watson (Ferrucci et al. 2010), the first machine learning system to win the quiz game Jeopardy! in 2011. More recently, Google observed substantial improvements building an ensemble-based neural machine translation system (Wu et al. 2016).

Our system consists of an ensemble of response models. The response models take as input a dialogue and output a response in natural language text. In addition, the response models may also output one or several scalar values, indicating their internal confidence. As will be explained later, the response models have been engineered to generate responses on a diverse set of topics using a variety of strategies. As input, the overall system expects to be given a dialogue history (i.e. all utterances recorded in the dialogue so far, including the current user utterance) and confidence values of the automatic speech recognition system (ASR confidences). To generate a response, the system follows

[2] Throughout the semi-finals we carried out several A/B testing experiments to evaluate different variants of our system (see Section 5). The score 3.15 is based on the best performing system in the period between July 29th and August 6th, 2017. The score is not based on the leaderboard which averages the scores of all the variants of our system (including a supervised learning system and a heuristic baseline system).

[3] The competition rules forbid us from stating the performance of any other team.

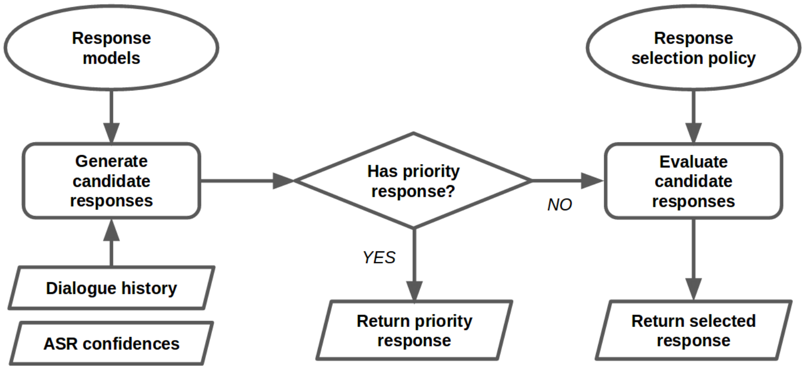


Figure 1: Dialogue manager control flow.

a three-step procedure. First, it uses all response models to generate a set of candidate responses. Second, if there exists a *priority* response in the set of candidate responses (i.e. a response which takes precedence over other responses), this response will be returned by the system.[4] For example, for the question *"What is your name?"*, the response *"I am an Alexa Prize Socialbot"* is a priority response. Third, if there are no *priority* responses, the response is selected by the *model selection policy*. For example, the *model selection policy* may select a response by scoring all candidate responses and picking the highest-scored response. The overall process is illustrated in Figure 1.

When the ASR confidences are below a certain threshold, the system requests the user to repeat their last utterance. Otherwise, the system does not utilize the ASR confidences. In particular, neither the response models nor the *model selection policy* make use of the ASR confidences. Nevertheless, the ASR system is far from perfect. Therefore, it is very plausible that the system could be improved significantly by conditioning the response models and *model selection policy* on the ASR confidences.

3 Response Models

There are 22 response models in the system, including retrieval-based neural networks, generation-based neural networks, knowledge base question answering systems and template-based systems. Examples of candidate model responses are shown in Table 1. This section aims to summarize these response models. First, the template-based models are described.

3.1 Template-based Models

We start by describing the template-based response models in the system.

Alicebot: *Alicebot* uses a set of AIML (artificial intelligence markup language) templates to produce a response given the dialogue history and user utterance (Wallace 2009, Shawar & Atwell 2007). We use the freely available Alice kernel available at `www.alicebot.org`. By default all templates generate non-priority responses, so we configure templates related to the socialbot's name, age and location to output priority responses. We modify a few templates further to make them consistent with the challenge (e.g. to avoid obscene language and to encourage the user to discuss certain topics, such as news, politics and movies). The majority of templates remain unchanged.

The *Alicebot* model also outputs a scalar confidence score. Since the AIML templates repeat the user's input utterance, they are not always correct sentences. Therefore, we use a string-based rules to determine if the response constitutes a correct sentence. If the response is correct sentence, it returns a high

[4] An ordering of the models decides which response to return in case there are multiple *priority* responses.

Table 1: Example dialogues and corresponding candidate responses generated by response models. The response of the final system is marked in bold.

Dialogue	**Dialogue**
SYSTEM: Hi. This is an Alexa Prize socialbot. USER: Hi SYSTEM: Do you have pets? USER: Yes, I have two rabbits.	... USER: Let's talk about movies. SYSTEM: What's your favorite movie? USER: star wars movie
Candidate Responses	**Candidate Responses**
BoWEscapePlan: I don't have an answer for this. BoWFactGenerator: Did you know that male rabbits are called bucks, females are does? GRUQuestionGenerator: How about the American family of it? Initiatorbot: Do you believe in love at first sight? LSTMClassifierMSMarco: Mar 23, 2017 Two Parts: Learning What to Feed Your Rabbit Learning How to ... DualEncoderRedditNews: bye doggie DualEncoderRedditPolitics: Who's edger u shboys? **Alicebot: Hurrah! Two is a good number of rabbits.** Elizabot: OK, but can you elaborate a bit? VHREDRedditMovies: Hmm. Nope, never heard of it. VHREDSubtitles: Would you like to see them? HelpResponse: I would really like to talk about news, politics or movies.	BoWEscapePlan: I don't have an answer for this. BoWFactGenerator: Here's an interesting fact. The international telephone dialing code for Antarctica is 672. GRUQuestionGenerator: how about your New York City? Initiatorbot: How was your day? LSTMClassifierMSMarco: A third Anthology film will be released in 2020. DualEncoderRedditNews: Now whisper it to me , one more time bby. DualEncoderRedditPolitics: hahaha on mobile this ain't even close. I wish I could show you but this looks awful I'm sorry man. Alicebot: What did you like about the robots in that movie? Elizabot: How does that make you feel? **Evi: Star Wars movie a movie in the Star Wars series.** VHREDRedditMovies: Oh please. Please. Pleeeease. Let this happen. VHREDSubtitles: What? HelpResponse: I would really like to talk about news, politics or movies.

confidence and otherwise it returns a low confidence score. This process is illustrated in Algorithm 1.

Algorithm 1: Alicebot

```
input: dialogue history
response ← apply AIML templates to dialogue history
if response is correct sentence then
    if response is given priority then
        confidence ← 1.0
    else
        confidence ← 0.5
else
    confidence ← 0.0
output: response, priority, confidence
```

Elizabot Similar to *Alicebot*, the *Elizabot* model performs string matching to select an answer from a set of templates. The model is based on the famous Eliza system, designed to mimic a Rogerian psychotherapist. (Weizenbaum 1966).[5] Therefore, in contrast with *Alicebot*, most of *Elizabot*'s responses are personal questions which are meant to engage the user to continue the conversation.

[5] We use the implementation available at: `https://gist.github.com/bebraw/273706`.

Here are two example templates:

1. *"I am (.*)" → "Did you come to me because you are ..."*
2. *"What (.*)" → "Why do you ask?"*

The ellipses mark the parts of the response sentence which will be replaced with text from the user's utterance. The model detects the appropriate template and selects the corresponding response (if there are multiple templates, then a template is selected at random). The model then runs the template response through a set of *reflections* to better format the string for a response (e.g. *"I'd" → "you would"*, *"your" → "my"*).

Algorithm 2: Initiatorbot

```
input: dialogue history
if Initiatorbot was triggered in one of last two turns then
    return ""
else if user did not give a greeting then
    return a non-priority response with a random initiator phrase
else
    return a priority response with a random initiator phrase
```

Initiatorbot The *Initiatorbot* model acts as a *conversation starter*: it asks the user an open-ended question to get the conversation started and increase the engagement of the user. We wrote 40 question phrases for the *Initiatorbot*. Examples of phrases include *"What did you do today?"*, *"Do you have pets?"* and *"What kind of news stories interest you the most?"*. As a special case, the model can also start the conversation by stating an interesting fact. In this case, the initiator phrase is *"Did you know that <fact>?"*, where *fact* is replaced by a statement. The set of facts is the same as used by the *BoWFactGenerator* model, described later.

Before returning a response, *Initiatorbot* first checks that it hasn't already been triggered in the last two turns of the conversation. If the user gives a greeting (e.g. *"hi"*), then *Initiatorbot* will return a response with priority. This is important because we observed that greetings often indicate the beginning of a conversation, where the user does not have a particular topic they would like to talk about. By asking a question, the system takes the *initiative* (i.e. control of the dialogue). The procedure is detailed in Algorithm 2.

Storybot The *Storybot* model outputs a short fiction story at the request of the user. We implemented this model as we observed that many users were asking the socialbot to tell stories.[6] *Storybot* determines if the user requested a story by checking if there was both a request word (e.g. *say*, *tell*.) and story-type word in the utterance (e.g. *story*, *tale*). The response states the story's title and author followed by the story body. For example, one set of responses from this model follows the pattern *"Alright, let me tell you the story <story_title> <story_body> by <story_author>"* where <story_title> is the title of the story, <story_body> is the main text and <story_author> is the name of the story's author. The stories were scraped from the website: `www.english-for-students.com`.

An example story is:

> **** The Ant and The Grasshopper ****
>
> The ants worked hard in summer. They sorted food for winter.
>
> At that time, a grasshopper remained idle. When winter came, the ants had enough to eat.
>
> But, the grasshopper had nothing to eat. He had to starve.
>
> He went to the ants and begged for foods. The ants asked in return, "What did you do in summer?"
>
> He replied, "I idled away my time during summer".
>
> The ant replied, "Then you must starve in winter." MORAL: Never be idle.

The *Storybot* is the only component in the system performing a *non-conversational activity*. It is triggered only when a user specifically asks for a story, and in that case its response is a priority

[6] Requests for telling stories is possibly a side-effect of user's interacting with bots from other teams, which often emphasized *non-conversational activities*, such as telling stories and playing quizzes and word games.

response. Otherwise, the *Storybot* response model is never triggered. Further, the rest of the system will not encourage the user to request stories.

3.2 Knowledge Base-based Question Answering

Evibot The *Evibot* response model forwards the user's utterance to Amazon's question-answering web-service *Evi*: `www.evi.com`. *Evi* was designed primarily to handle factual questions. Therefore, *Evibot* returns a priority response for direct questions, defined as user utterances containing a wh-word (e.g. *"who"*, *"what"*), and otherwise returns a non-priority or, possibly, an empty response. If the query is a direct question and contains non-stop words, *Evibot* will follow a three step procedure to generate its response. First, *Evibot* forwards a query to `www.evi.com` containing the whole user utterance, and returns the resulting answer if its valid. If that fails, *Evibot* applies NLTK's named entity processor (Bird et al. 2009) to the query to find subqueries with named entities. For each subphrase that contains a named entity, *Evibot* forwards queries to `www.evi.com`, and returns the result upon a valid response. Finally, if the previous two steps fail, *Evibot* forwards queries for every subquery without named entities, and returns either a valid response or an empty response. The procedure is detailed in Algorithm 3.

Algorithm 3: Evibot

```
input: dialogue history
query ← last user utterance
has-wh-words ← true if utterance contains a wh-word, otherwise false
has-only-stop-words ← true if utterance only has stop words, otherwise false
if has-only-stop-words and not has-wh-words then
    return ""
evi-response ← send query to www.evi.com
priority ← true if has-wh-words and evi-response is valid, otherwise false
if evi-response is valid then
    return evi-response, priority
else if has-wh-words then
    priority ← has-wh-words
    subentities ← entities extracted from query using NLTK's named entity processor
    subphrases ← list of subphrases with entities
    for subphrase in subphrases do
        evi-response ← send subphrase to www.evi.com
        if evi-response is valid then
            return evi-response, priority
    subphrases ← list of all subphrases
    for subphrase in subphrases do
        evi-response ← send subphrase to www.evi.com
        if evi-response is valid then
            return evi-response, priority
else
    return ""
```

BoWMovies The *BoWMovies* model is a template-based response model, which handles questions in the movie domain. The model has a list of entity names and tags (e.g. *movie plot* and *release year*). The model searches the user's utterance for known entities and tags. Entities are identified by string matching. This is done in a cascading order, by giving first preference to movie title matches, then actor name matches, and finally director name matches. Tags are also identified by string matching. However, if exact string matching fails for tags, then identification is performed by word embedding similarity. If both an entity and a tag are present, the agent will dispatch an API call to one of several data sources to retrieve the data item for the selected query type. The agent is limited by the data available in the APIs to which it has access. The model's responses follow predefined templates.

Movie titles, actor names, and director names are extracted from the Internet Movie Database (IMDB). Movie descriptions are taken from Google Knowledge Graph's API. Other movie title queries are directed to the Open Movie Database (OMDB).[7] For actor and director queries, the Wikiedata API is used. First, a search for actor and director names is done on a Wikidata JSON dump.

As described earlier, the model uses word embeddings to match tags. These word embeddings are trained using Word2Vec on movie plot summaries and actor biographies extracted from the IMDB database (Mikolov et al. 2013).

Algorithm 4: BoWMovies - ComputeResponse

```
input: dialogue history
entity ← entity contained both in last user utterance and list of movie titles, actors or directors
if no entity then
    entity ← entity contained in previous user utterances and movie titles, actors or directors
if no entity then
    return ""
if entity is a movie title then
    response ← ComputeEntityResponse(entity, movie title)
else if entity is an actor name then
    response ← ComputeEntityResponse(entity, actor name)
else if entity is an director name then
    response ← ComputeEntityResponse(entity, director name)
return response
```

Algorithm 5: BoWMovies - ComputeEntityResponse

```
input: entity and entity type
tag ← string matching tag, where tag is valid for entity type (movie title, actor name, director name)
if no tag then
    tag ← word embedding matching tag, where tag is a single word and valid for the entity type (movie title, actor name, director name)
if no tag then
    tag ← word embedding matching tag, where tag is multiple words and valid for the entity type (movie title, actor name, director name)
if no tag then
    return ""
api-response ← call external API with query (entity, tag).
response ← template with api-response inserted
return response
```

3.3 Retrieval-based Neural Networks

VHRED models: The system contains several VHRED models, sequence-to-sequence models with Gaussian latent variables trained as variational auto-encoders (Serban et al. 2017, Kingma & Welling 2014, Rezende et al. 2014). The models are trained using the same procedure as Serban et al. (2017). The trained VHRED models generate candidate responses as follows. First, a set of K model responses are retrieved from a dataset using cosine similarity between the current dialogue history and the dialogue history in the dataset based on bag-of-words TF-IDF Glove word embeddings (Pennington et al. 2014).[8] An approximation of the log-likelihood for each of the 20 responses is computed by VHRED, and the response with the highest log-likelihood is returned. The system has 4 VHRED models based on datasets scraped from Reddit, one VHRED model based on news articles and one VHRED model based on movie subtitles:

- *VHREDRedditPolitics* trained on `https://www.reddit.com/r/politics` and extracting responses from all Reddit datasets with $K = 10$,

[7] See `www.omdbapi.com`. This should not be confused with IMDB.

[8] We use the Glove embeddings trained on Wikipedia 2014 + Gigaword 5: `https://nlp.stanford.edu/projects/glove/`.

- *VHREDRedditNews* trained on Reddit `https://www.reddit.com/r/news` and extracting responses from all Reddit datasets with $K = 20$,
- *VHREDRedditSports* trained on Reddit `https://www.reddit.com/r/sports` and extracting responses from all Reddit datasets with $K = 20$,
- *VHREDRedditMovies* trained on Reddit `https://www.reddit.com/r/movies` and extracting responses from all Reddit datasets with $K = 20$,
- *VHREDWashingtonPost*[9] trained on Reddit `https://www.reddit.com/r/politics` and extracting responses from user comments to WashingtonPost news articles, and
- *VHREDSubtitles*[10] using the movie subtitles dataset SubTle (Ameixa et al. 2014) with $K = 10$.

In particular, *VHREDRedditPolitics* and *VHREDWashingtonPost* use a different retrieval procedure. These two models use a logistic regression model to score the responses instead of the approximate log-likelihood. The logistic regression model is trained on a set of 7500 Reddit threads and candidate responses annotated by Amazon Mechanical Turk workers on a Likert-type scale $1-5$. The candidate responses are selected from other Reddit threads according to cosine similarity w.r.t. Glove word embeddings. The label collection and training procedure for the logistic regression model are similar to the procedures described in Section 4. For each response, the logistic regression model takes as input the VHRED log-likelihood score, as well as several other input features, and outputs a scalar-valued score. Even though the logistic regression model did improve the appropriateness of responses selected for Reddit threads, *VHREDRedditPolitics* is used extremely rarely in the final system (see Section 4). This suggests that training a model to rerank responses based on labeled Reddit threads and responses cannot help improve performance.

SkipThought Vector Models: The system contains a SkipThought Vector model (Kiros et al. 2015) trained on the BookCorpus dataset (Zhu et al. 2015) and on the SemEval 2014 Task 1 (Marelli et al. 2014). The model was trained using the same procedure as Kiros et al. (2015) and is called *SkipThoughtBooks*.

SkipThoughtBooks ensures that the system complies with the Amazon Alexa Prize competition rules. One rule, introduced early in the competition, is that socialbots were not supposed to state their own opinions related to political or religious topics. If a user wishes to discuss such topics, the socialbots should proceed by asking questions or stating facts. *SkipThoughtBooks* also handles idiosyncratic issues particular to the Alexa platform. For example, many users did not understand the purpose of a socialbot and asked our socialbot to play music. In this case, the system should instruct the user to exit the *socialbot application* and then play music.

SkipThoughtBooks follows a two-step procedure to generate its response. The first step compares the user's last utterance to a set of trigger phrases. If a match is found, the model returns a corresponding priority response.[11] For example, if the user says *"What do you think about Donald trump?"*, the model will return a priority response, such as *"Sometimes, truth is stranger than fiction."*. A match is found if: 1) the SkipThought Vector model's semantic relatedness score between the user's last utterance and a trigger phrase is above a predefined threshold, and 2) the user's last utterance contains keywords relevant to the trigger phrase.[12] In total, there are 315 trigger phrases (most are paraphrases of each other) and 35 response sets.

If the model did not find a match in the first step, it proceeds to the second step. In this step, the model selects its response from among all Reddit dataset responses. As before, a set of K model responses are retrieved using cosine similarity. The model then returns the response with the highest semantic relatedness score.

Dual Encoder Models: The system contains two Dual Encoder retrieval models (Lowe et al. 2015), *DualEncoderRedditPolitics* and *DualEncoderRedditNews*. Both models are composed of two sequence encoders ENC$_Q$ and ENC$_R$ with a single LSTM recurrent layer used to encode the dialogue history and a candidate response. The score for a candidate response is computed by a bilinear mapping of the dialogue history embedding and the candidate response embedding as Lowe et al. (2015).

[9] For *VHREDWashingtonPost*, the K responses are extracted based on the cosine similarity between the current dialogue and the news article keywords. K varies depending on the number of user comments within a set of news articles above a certain cosine similarity threshold.

[10] For *VHREDSubtitles*, cosine similarity is computed based on one-hot vectors for each word.

[11] Trigger phrases may have multiple responses. In this case, a response is selected at random.

[12] Some trigger phrases do not have keywords. In this case, matching is based only on semantic relatedness.

The models are trained using the method proposed by (Lowe et al. 2015). The response with the highest score from a set of $K = 50$ candidate responses are retrieved using TF-IDF cosine similarity based on Glove word embeddings. The model *DualEncoderRedditPolitics* is trained on the Reddit `https://www.reddit.com/r/politics` dataset and extracts responses from all Reddit datasets. The model *DualEncoderRedditNews* is trained on the Reddit `https://www.reddit.com/r/news` dataset and extracts responses from all Reddit datasets.

Bag-of-words Retrieval Models: The system contains three bag-of-words retrieval models based on TF-IDF Glove word embeddings (Pennington et al. 2014) and Word2Vec embeddings (Mikolov et al. 2013).[13] Similar to the VHRED models, these models retrieve the response with the highest cosine similarity. The *BoWWashingtonPost* model retrieves user comments from WashingtonPost news articles using Glove word embeddings. The model *BoWTrump* retrieves responses from a set of Twitter tweets scraped from Donald Trump's profile: `https://twitter.com/realDonaldTrump`. This model also uses Glove word embeddings and it only returns a response when at least one relevant keyword or phrase is found in the user's utterance (e.g. when the word *"Trump"* is mentioned by the user). The list of trigger keywords and phrases include: *'donald'*, *'trump'*, *'potus'*, *'president of the united states'*, *'president of the us'*, *'hillary'*, *'clinton'*, *'barack'*, and *'obama'*. The model *BoWFactGenerator* retrieves responses from a set of about 2500 *interesting* and *fun* facts, including facts about animals, geography and history. The model uses Word2Vec word embeddings. The model *BoWGameofThrones* retrieves responses from a set of quotes scraped from `https://twitter.com/ThroneQuotes` using Glove word embeddings. Tweets from this source were manually inspected and cleaned to remove any tweets that were not quotes from the series. As in the *BoWTrump* model, we use a list of trigger phrases to determine if the model's output is relevant to the user's utterance. We populate this list with around 80 popular character names, place names and family names, which are large unique to the domain. We also added a few aliases to try and account for alternative speech transcriptions of these named entities. Some phrases include: *'ned stark'*, *'jon snow'*, *'john snow'*, *'samwell tarly'*, *"hodor"*, *"dothraki"* and so on. [14]

3.4 Retrieval-based Logistic Regression

BoWEscapePlan: The system contains a response model, called *BoWEscapePlan*, which returns a response from a set of 35 topic-independent, generic pre-defined responses, such as *"Could you repeat that again"*, *"I don't know"* and *"Was that a question?"*. Its main purpose is to maintain user engagement and keep the conversation going, when other models are unable to provide meaningful responses. This model uses a logistic regression classifier to select its response based on a set of higher-level features.

To train the logistic regression classifier, we annotated $12,000$ user utterances and candidate response pairs for appropriateness on a Likert-type scale $1 - 5$. The user utterances were extracted from interactions between Alexa users and a preliminary version of the system. The candidate responses were sampled at random from *BoWEscapePlan*'s response list. The label collection and training procedure for the logistic regression model are similar to the procedures described in Section 4. The logistic regression model is trained with log-likelihood on a training set, with early-stopping on a development set, and evaluated on the testing set. However, the trained model's performance was poor. It obtained a Pearson correlation coefficient of 0.05 and a Spearman's rank correlation coefficient of 0.07. This indicates that the logistic regression model is only slightly better at selecting a topic-independent, generic response compared to selecting a response at uniform random. Future work should investigate collecting more labeled data and pre-training the logistic regression model.

3.5 Search Engine-based Neural Networks

The system contains a deep classifier model, called *LSTMClassifierMSMarco*, which chooses its response from a set of search engine results. The system searches the web with the last user utterance as query, and retrieves the first 10 search snippets. The retrieved snippets are preprocessed by stripping trailing words, removing unnecessary punctuation and truncating to the last full sentence. The model uses a bidirectional LSTM to separately map the last dialogue utterance and the snippet to their own embedding vectors. The resulting two representations are concatenated and passed

[13] We use the pre-trained Word2Vec embeddings: `https://code.google.com/archive/p/word2vec/`.

[14] This model was implemented after the competition ended, but is included here for completeness.

through an MLP to predict a scalar-value between $0 - 1$ indicating how appropriate the snippet is as a response to the utterance.

The model is trained as a binary classification model on the Microsoft Marco dataset with cross-entropy to predict the relevancy of a snippet given a user query (Nguyen et al. 2016). Given a search query and a search snippet, the model must output one when the search snippet is relevant and otherwise zero. Search queries and ground truth search snippets are taken as positive samples, while other search snippets are selected at random as negative samples. On this task, the model is able to reach a prediction accuracy of 72.96% w.r.t. the Microsoft Marco development set.

The system is able to use search APIs from various search engines including Google, Bing, and AIFounded (Im 2017). In the current model, we choose Google as the search engine, since qualitative inspection showed that this retrieved the most appropriate responses.

3.6 Generation-based Neural Networks

The system contains a generative recurrent neural network language model, called *GRUQuestion-Generator*, which can generate follow-up questions word-by-word, conditioned on the dialogue history. The input to the model consists of three components: a one-hot vector of the current word, a binary question label and a binary speaker label. The model contains two GRU layers (Cho et al. 2014) and softmax output layer. The model is trained on Reddit Politics and Reddit News conversations, wherein posts were labeled as questions by detecting question marks. We use the optimizer Adam (Kingma & Ba 2015), and perform early stopping by checking the perplexity on the validation set For generation, we first condition the model on a short question template (e.g. *"How about"*, *“What about”*, *“How do you think of”*, *“What is your opinion of”*), and then generate the rest of the question by sampling from the model with the question label clamped to one. The generation procedure stops once a question mark is detected. Further, the length of the question is controlled by tuning the temperature of the softmax layer. Due to speed requirements, only two candidate responses are generated and the best one w.r.t. log-likelihood of the first 10 words is returned.

4 Model Selection Policy

After generating the candidate response set, the dialogue system uses a *model selection policy* to select the response it returns to the user. The system must select a response which increases the satisfaction of the user for the entire dialogue. It must make a trade-off between immediate and long-term user satisfaction. For example, suppose the user asks to talk about politics. If the system chooses to respond with a political joke, the user may be pleased for one turn. Afterwards, however, the user may be disappointed with the system's inability to debate political issues. Instead, if the system chooses to respond with a short news story, the user may be less pleased for one turn. However, the news story may influence the user to follow up with factual questions, which the system may be better adept at handling. To make the trade-off between immediate and long-term user satisfaction, we consider selecting the appropriate response as a *sequential decision making problem.* This section describes five approaches to learn the model selection policy. These approaches are all evaluated with real-world users in the next section.

We use the reinforcement learning framework (Sutton & Barto 1998). The system is an agent, which takes actions in an environment in order to maximize rewards. For each time step $t = 1, \dots, T$, the agent observes the dialogue history h_t and must choose one of K actions (responses): $a_t^1, \dots, a_t^K$. After taking an action, the agent receives a reward r_t and is transferred to the next state h_{t+1} (which includes the user's next response). Then, the agent is provided with a new set of K actions: $a_{t+1}^1, \dots, a_{t+1}^K$. The agent's goal is to maximize the discounted sum of rewards:

$$R = \sum_{t=1}^{T} \gamma^t r_t, \tag{1}$$

which is referred to as the *expected cumulative return* (or simply expected return). The parameter $\gamma \in (0, 1]$ is a discount factor.

An issue specific to our setting is that the set of actions changes depending on the state (dialogue history). This happens because the candidate responses are generated by response models, which also depend on the dialogue history. In addition, the response models are not deterministic. This means the set of candidate responses is likely to be different every time the agent encounters the same state h_t.[15] This is in contrast to certain reinforcement learning problems, such as learning to play Atari 2600 games, where the set of actions is fixed given the state. To simplify notation, we will fix the number of actions to K henceforth.

Action-value Parametrization: We use two different approaches to parametrize the agent's policy. The first approach is based on an action-value function, defined by parameters θ:

$$Q_\theta(h_t, a_t^k) \in \mathbb{R} \quad \text{for } k = 1, \ldots, K, \tag{2}$$

which estimates expected return of taking action a_t^k (candidate response k) given dialogue history h_t and given that the agent will continue to use the same policy afterwards. Given Q_θ, the agent chooses the action with highest expected return:

$$\pi_\theta(h_t) = \arg\max_k \; Q_\theta(h_t, a_t^k). \tag{3}$$

The use of an action-value function for selecting dialogue responses is closely related to the recent work by Lowe et al. (2017), where a model is learned to predict the quality of a dialogue system response. However, in our case, Q_θ is only conditioned on the dialogue context. On the other hand, the model proposed by Lowe et al. (2017) is conditioned both on the dialogue context and on a human reference response. The action-value function is also related to the the work by Yu et al. (2016), who learn an evaluation model, which is used to train a reinforcement learning agent to select appropriate dialogue response strategies.

Stochastic Policy Parametrization: The second approach parametrizes the policy as a discrete distribution over actions. Let θ be the parameters. The agent selects its action by sampling:

$$\pi_\theta(a_t^k | h_t) = \frac{e^{\lambda^{-1} f_\theta(h_t, a_t^k)}}{\sum_{a'_t} e^{\lambda^{-1} f_\theta(h_t, a'_t)}} \quad \text{for } k = 1, \ldots, K, \tag{4}$$

where $f_\theta(h_t, a_t^k)$ is the *scoring function*, which assigns a scalar score to each response a_t^k given h_t. The parameter λ is called the temperature and controls the entropy of the distribution. The higher λ is, the more uniform the selecting of actions will be. The stochastic policy can be transformed to a deterministic (greedy) policy by selecting the action with highest probability:

$$\pi_\theta^{\text{greedy}}(h_t) = \arg\max_k \; \pi_\theta(a_t^k | h_t) = \arg\max_k \; f_\theta(h_t, a_t^k). \tag{5}$$

Scoring Model: The action-value function $Q_\theta(h_t, a_t^k)$ and scoring function $f_\theta(h_t, a_t^k)$ are closely related. Both functions yield a ranking over the actions; higher values imply higher expected returns. When $Q_\theta(h_t, a_t^k) = f_\theta(h_t, a_t^k)$, the action-value function policy in eq. (3) is equivalent to the greedy policy in eq. (5). For simplicity, we will use the same parametrization for both $Q_\theta(h_t, a_t^k)$ and $f_\theta(h_t, a_t^k)$. Therefore, we let both functions take the same features as input and process them using the same neural network architecture. We will refer to both functions as the *scoring model.*

The next section describes the input features for the scoring model.

4.1 Input Features

As input to the scoring model we compute 1458 features based on the given dialogue history and candidate response. The input features are based on a combination of word embeddings, dialogue acts, part-of-speech tags, unigram word overlap, bigram word overlap and model-specific features:

Word embeddings of response: Average of candidate response word embeddings (Mikolov et al. 2013).[16]

Word embeddings of last user utterance: Average of the last user utterance word embeddings.

Word embeddings of context: Average of the word embeddings of the last six utterances in dialogue context.

[15] In general, since some response models only output responses for certain user utterances, the number of candidate responses also changes depending on the state.

[16] We use the pre-trained Word2Vec embeddings: `https://code.google.com/archive/p/word2vec/`.

Word embedding of user context: Average of the word embeddings of the last three user utterances in dialogue context.

Word embedding similarity metrics: The *Embedding Average*, *Embedding Extrema* and *Embedding Greedy* similarity metrics described by Liu et al. (2016). Each similarity metric is computed between 1) the last user utterance and candidate response, 2) the last six utterances in the dialogue and candidate response, 3) the last three user utterances in the dialogue and candidate response, 4) the last six utterances in the dialogue and candidate response with stop-words removed, and 5) the last three user utterances in the dialogue and candidate response with stop-words removed.

Response model class: A one-hot vector with size equal to the number of response models, where entry i is equal to 1.0 when candidate response was generated by the model class with index i.

Part-of-speech response class: The part-of-speech tags for candidate response is estimated using a maximum entropy tagger trained on the Penn Treebank corpus. The sequence of part-of-speech tags is then mapped to a one-hot vector, which constitutes the input feature.

Dialogue act response model class: The outer-product between a one-hot vector representing the dialogue act (we consider 10 types of dialogue acts) and a one-hot vector for indicating the model class (Stolcke et al. 2000).

Word overlap: 1.0 when one or more non-stop-words overlap between candidate response and last user utterance, and otherwise zero.

Bigram overlap short-term: 1.0 when a bigram (two consecutive tokens) exists both in the candidate response and in the last user utterance, and otherwise zero.

Bigram overlap long-term: 1.0 when a bigram exists both in candidate response and in one of the last utterances in dialogue context, and otherwise zero.

Named-entity overlap short-term: 1.0 when a named-entity (an upper-cased word, which is not a stop-word) exists both in candidate response and in the last user utterance, and otherwise zero.

Named-entity overlap long-term: 1.0 when a named-entity exists both in candidate response and in one of the last utterances in dialogue context, and otherwise zero.

Generic response: 1.0 when candidate response consists of only stop-words or words shorter than 3 characters, and otherwise zero.

Wh-word response feature: 1.0 when candidate response contains a wh-word (e.g. *what*, *where*, and so on), and otherwise zero.

Wh-word context: 1.0 when last user utterance contains a wh-word, and otherwise zero.

Intensifier word response: 1.0 when candidate response contains an intensifier word (e.g. *amazingly*, *crazy*, and so on), and otherwise zero.

Intensifier word context: 1.0 when last user utterance contains an intensifier word, and otherwise zero.

Unigram response: A set of binary features which are 1.0 when candidate response contains a specific word (including the words *I*, *you* and *thanks*), and otherwise zero.

Negation response: 1.0 when candidate response contains a negation word, such as *not* or *n't*, and otherwise zero.

Non-stop-words response: 1.0 when candidate response contains a non-stop-word, and otherwise zero.

We do not include features based on the confidences of the speech recognition system, for experimental reasons. Speech recognition errors are a confounding factor in experiments with real-world users. Speech recognition errors are likely to affect user satisfaction. If features based on speech recognition confidences were included, one policy might learn to handle speech recognition errors better than another policy. In turn, this could make that policy perform better w.r.t. overall user satisfaction. However, that would be an effect caused by the imperfect speech recognition system, and would not reflect user satisfaction under a perfect speech recognition system. Excluding these features as input to the scoring model helps minimize this confounding effect.Nevertheless, even if these features are excluded, it should be noted that speech recognition errors still constitute a substantial confounding factor in our later experiments. Lastly, for the same reasons, none of the response models utilize speech recognition confidences.

In principle, it is possible to compute input features by encoding the dialogue context and candidate response using Recurrent Neural Networks (RNNs) or Convolutional Neural Networks (ConvNets) (Socher et al. 2013, Blunsom et al. 2014, Cho et al. 2014, Yu et al. 2014, Kiros et al. 2015). However, these models are known to require training on large corpora in order to achieve acceptable performance, which we do not have access to. In addition, we need to keep the scoring model's execution time under 150ms. Otherwise, the slowdown in the response time, could frustrate the user and lower the overall user satisfaction. This rules out large RNNs and ConvNets for the Amazon Alexa Prize competition, since these would require more computational runtime. However, future dialogue systems utilizing larger datasets should consider large-scale models.

4.2 Model Architecture

This section describes the scoring model's architecture. The scoring model is a five-layered neural network. The first layer is the input, consisting of the 1458 features, described in the previous section. The second layer contains 500 hidden units, computed by applying a linear transformation followed by the rectified linear activation function (Nair & Hinton 2010, Glorot et al. 2011) to the input layer units. The third layer contains 20 hidden units, computed by applying a linear transformation to the preceding layer units. Similar to matrix factorization, this layer compresses the 500 hidden units down to 20 hidden units. The fourth layer contains 5 outputs units, which are probabilities (i.e. all values are positive and sum to one). These output units are computed by applying a linear transformation to the preceding layer units followed by a softmax transformation. This layer corresponds to the Amazon Mechanical Turk labels, which will be described in the next sub-section. The fifth layer is the final output scalar, computed by applying a linear transformation to the units in the third and fourth layers. The model is illustrated in Figure 2.

Before settling on this architecture, we experimented both with deeper and more shallow models. However, we found that both the deeper models and the more shallow models performed worse. Nevertheless, future work should explore alternative architectures.

We use five different machine learning approaches to learn the scoring model. These are described next.

4.3 Supervised AMT: Learning with Crowdsourced Labels

This section describes the first approach to learning the scroing model, which is based on supervised learning from crowdsourced examples. This approach also serves as initialization for the approaches discussed later.

Crowdsourcing: We use Amazon Mechanical Turk (AMT) to collect data for training the scoring model. We follow a setup similar to Liu et al. (2016). We show human evaluators a dialogue along with 4 candidate responses, and ask them to score how appropriate each candidate response is on a

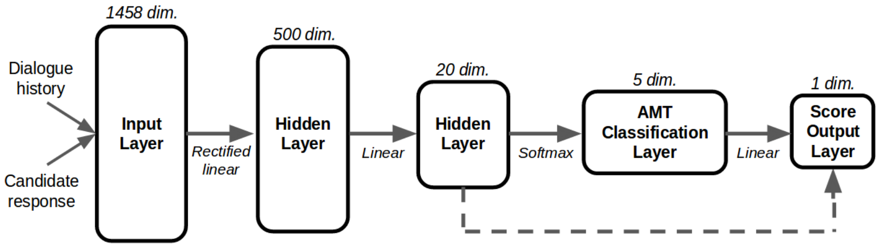


Figure 2: Computational graph for scoring model, used for the model selection policies based on both action-value function and stochastic policy parametrizations. The model consists of an input layer with 1458 features, a hidden layer with 500 hidden units, a hidden layer with 20 hidden units, a softmax layer with 5 output probabilities (corresponding to the five AMT labels in Section 4.3), and a scalar-valued output layer. The dashed arrow indicates a skip connection.

1-5 Likert-type scale. The score 1 indicates that the response is inappropriate or does not make sense, 3 indicates that the response is acceptable, and 5 indicates that the response is excellent and highly appropriate.

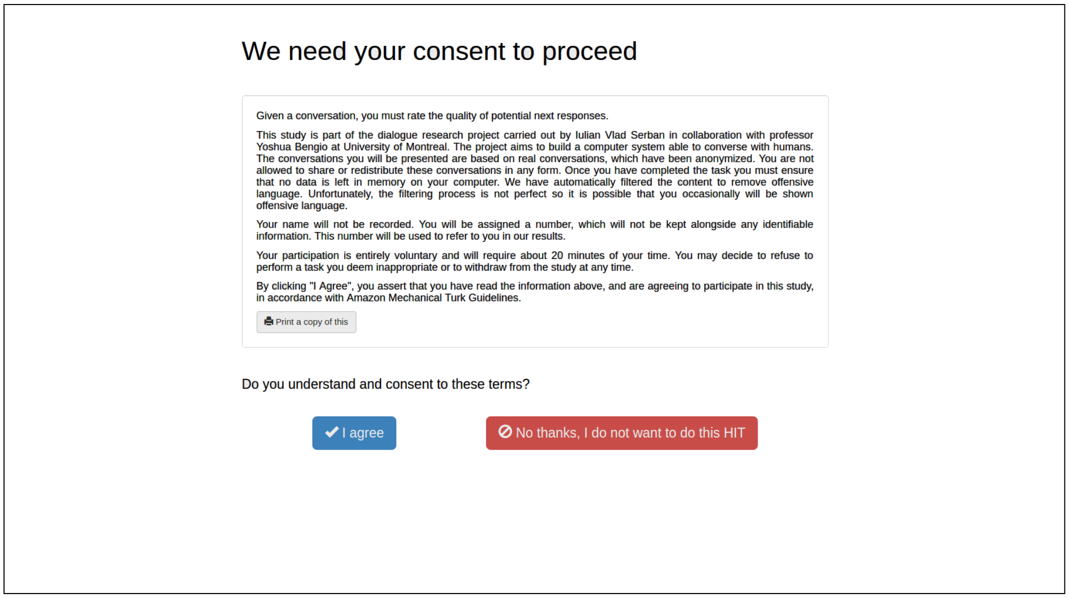
Our setup only asks human evaluators to rate the overall appropriateness of the candidate responses. In principle, we could choose to evaluate other aspects of the candidate responses. For example, we could evaluate fluency. However, fluency ratings would not be very useful since most of our models retrieve their responses from existing corpora, which contain mainly fluent and grammatically correct responses. As another example, we could evaluate topical relevancy. However, we choose not to evaluate such criteria since it is known to be difficult to reach high inter-annotator agreement on them (Liu et al. 2016). In fact, it is well known that even asking for a single overall rating tends to produce only a fair agreement between human evaluators (Charras et al. 2016); disagreement between annotators tends to arise either when the dialogue context is short and ambiguous, or when the candidate response is only partially relevant and acceptable.

The dialogues are extracted from interactions between Alexa users and preliminary versions of our system. Only dialogues where the system does not have a priority response were extracted (when there is a priority response, the dialogue manager must always return the priority response). About $3/4$ of these dialogues were sampled at random, and the remaining $1/4$ dialogues were sampled at random excluding identical dialogues.[17] For each dialogue, the corresponding candidate responses are created by generating candidate responses from the response models.

We preprocess the dialogues and candidate responses by masking out profanities and swear words with stars (e.g. we map *"fuck"* to *"****"*).[18] Furthermore, we anonymize the dialogues and candidate responses by replacing first names with randomly selected gender-neutral names (for example, *"Hi John"* could be mapped to *"Hello Casey"*). Finally, the dialogues are truncated to the last 4 utterances and last 500 words. This reduces the cognitive load of the annotators. Examples from the crowdsourcing task are shown in Figure 3, Figure 4 and Figure 5. The dialogue example shown in Figure 5 is a fictitious example.

[17] Sampling at random is advantageous for our goal, because it ensures that candidate responses to frequent user statements and questions tend to be annotated by more turkers. This increases the average annotation accuracy for such utterances, which in turn increases the scoring model's accuracy for such utterances.

[18] The masking is not perfect. Therefore, we also instruct turkers that the task may contain profane and obscene language. Further, it should also be noted that Amazon Mechanical Turk only employs adults.



We need your consent to proceed

Given a conversation, you must rate the quality of potential next responses.

This study is part of the dialogue research project carried out by Iulian Vlad Serban in collaboration with professor Yoshua Bengio at University of Montreal. The project aims to build a computer system able to converse with humans. The conversations you will be presented are based on real conversations, which have been anonymized. You are not allowed to share or redistribute these conversations in any form. Once you have completed the task you must ensure that no data is left in memory on your computer. We have automatically filtered the content to remove offensive language. Unfortunately, the filtering process is not perfect so it is possible that you occasionally will be shown offensive language.

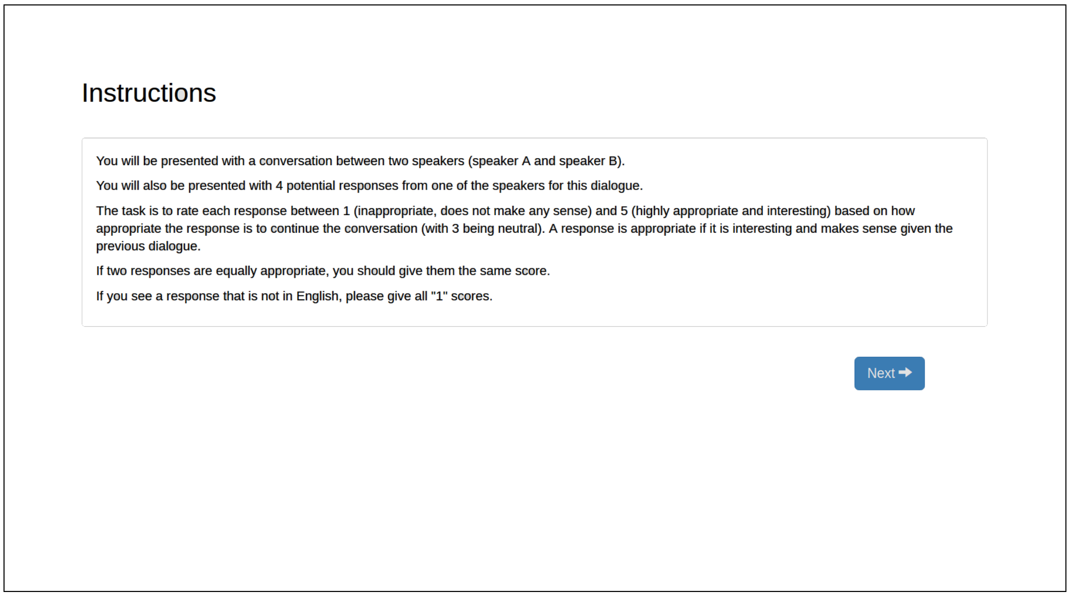
Your name will not be recorded. You will be assigned a number, which will not be kept alongside any identifiable information. This number will be used to refer to you in our results.

Your participation is entirely voluntary and will require about 20 minutes of your time. You may decide to refuse to perform a task you deem inappropriate or to withdraw from the study at any time.

By clicking "I Agree", you assert that you have read the information above, and are agreeing to participate in this study, in accordance with Amazon Mechanical Turk Guidelines.

Do you understand and consent to these terms?

Figure 3: Consent screen for Amazon Mechanical Turk human intelligence tasks (HITs).



Instructions

You will be presented with a conversation between two speakers (speaker A and speaker B).

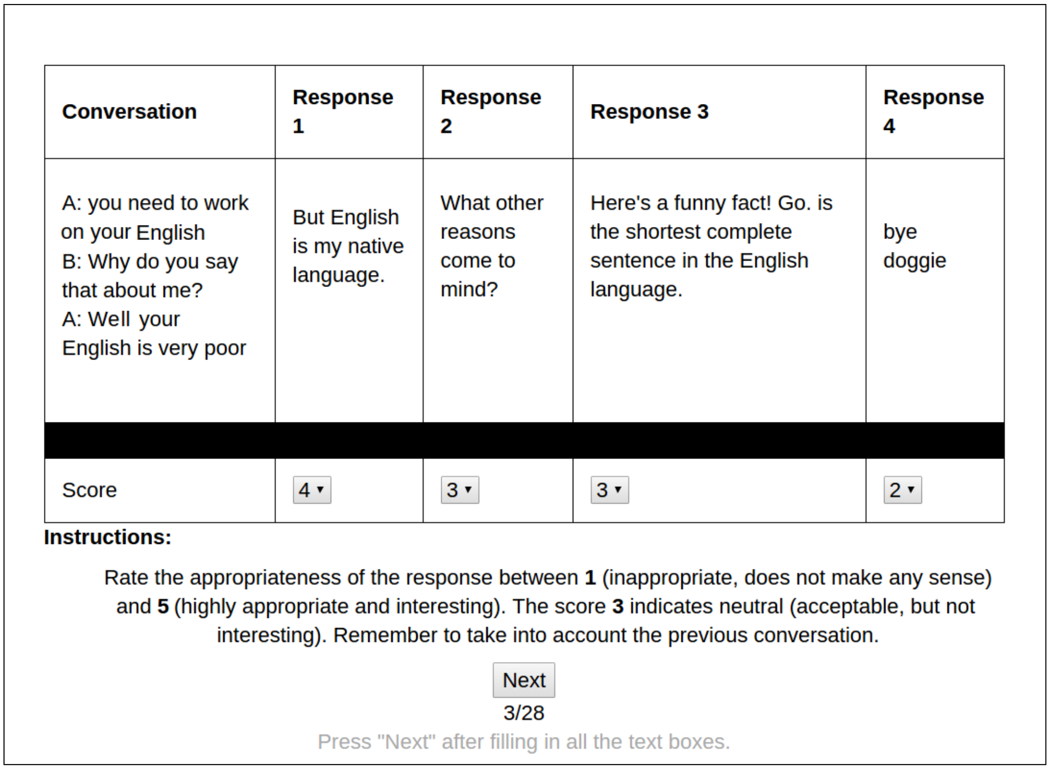
You will also be presented with 4 potential responses from one of the speakers for this dialogue.

The task is to rate each response between 1 (inappropriate, does not make any sense) and 5 (highly appropriate and interesting) based on how appropriate the response is to continue the conversation (with 3 being neutral). A response is appropriate if it is interesting and makes sense given the previous dialogue.

If two responses are equally appropriate, you should give them the same score.

If you see a response that is not in English, please give all "1" scores.

Figure 4: Instructions screen for Amazon Mechanical Turk human intelligence tasks (HITs).



Conversation	Response 1	Response 2	Response 3	Response 4
A: you need to work on your English B: Why do you say that about me? A: Well your English is very poor	But English is my native language.	What other reasons come to mind?	Here's a funny fact! Go. is the shortest complete sentence in the English language.	bye doggie
Score	4 ▾	3 ▾	3 ▾	2 ▾

Instructions:

Rate the appropriateness of the response between **1** (inappropriate, does not make any sense) and **5** (highly appropriate and interesting). The score **3** indicates neutral (acceptable, but not interesting). Remember to take into account the previous conversation.

3/28

Press "Next" after filling in all the text boxes.

Figure 5: Annotation screen for Amazon Mechanical Turk human intelligence tasks (HITs). The dialogue text is a fictitious example.

We inspected the annotations manually. We observed that annotators tended to frequently overrate topic-independent, generic responses. Such responses may be considered acceptable for a single turn in a conversation, but are likely to be detrimental when repeated over and over again. In particular, annotators tended to overrate responses generated by the response models *Alicebot*, *Elizabot*, *VHREDSubtitles* and *BoWEscapePlan*. Responses generated by these models are often acceptable or good, but the majority of them are topic-independent, generic sentences. Therefore, for these response models, we mapped all labels 5 (*"excellent"*) to 4 (*"good"*). Furthermore, for responses consisting of only stop-words, we decreased the labels by one level (e.g. 4 is mapped to 3). Finally, the *BoWMovies* response model suffered from a bug during the label collection period. Therefore, we decreased all labels given to *BoWMovies* responses to be at most 2 (*"poor"*).

In total, we collected $199,678$ labels. We split this into training (train), development (dev) and testing (test) datasets consisting of respectively 137,549, 23,298 and 38,831 labels each.

Training: We optimize the scoring model w.r.t. log-likelihood (cross-entropy) to predict the 4th layer, which represents the AMT label classes. Formally, we optimize the parameters θ:

$$\hat{\theta} = \arg\max_{\theta} \sum_{x,y} \log P_{\theta}(y|x), \tag{6}$$

where x are the input features, y is the corresponding AMT label class (a one-hot vector) and $P_{\theta}(y|x)$ is the model's predicted probability of y given x, computed in the second last layer of the scoring model. We use the first-order gradient-descent optimizer Adam (Kingma & Ba 2015) We experiment with a variety of hyper-parameters, and select the best hyper-parameter combination based on the log-likelihood of the dev set. For the first hidden layer, we experiment with layer sizes in the set: $\{500, 200, 50\}$. For the second hidden layer, we experiment with layer sizes in the set: $\{50, 20, 5\}$. We use L2 regularization on all model parameters, except for bias parameters. We experiment with L2 regularization coefficients in the set: $\{10.0, 1.0, 10^{-1}, \dots, 10^{-9}\}$ Unfortunately, we do not have labels to train the last layer. Therefore, we fix the parameters of the last layer to the vector $[1.0, 2.0, 3.0, 4.0, 5.0]$. In other words, we assign a score of 1.0 for the label *very poor*, a score of 2.0 for the label *poor*, a score of 3.0 for the label *acceptable*, a score of 4.0 for the label *good* and a score of 5.0 for the label *excellent*. As this model was trained on crowdsourced data from Amazon Mechanical Turk (AMT), we call this model *Supervised AMT*.

Table 2: Scoring model evaluation on Amazon Mechanical Turk test set w.r.t. Pearson correlation coefficient, Spearman's rank correlation coefficient and mean squared error.

Model	**Pearson**	**Spearman**	**Mean squared error**
Average Predictor	0.00	0.00	1.30
Supervised AMT	$\mathbf{0.40}$	$\mathbf{0.38}$	$\mathbf{1.10}$

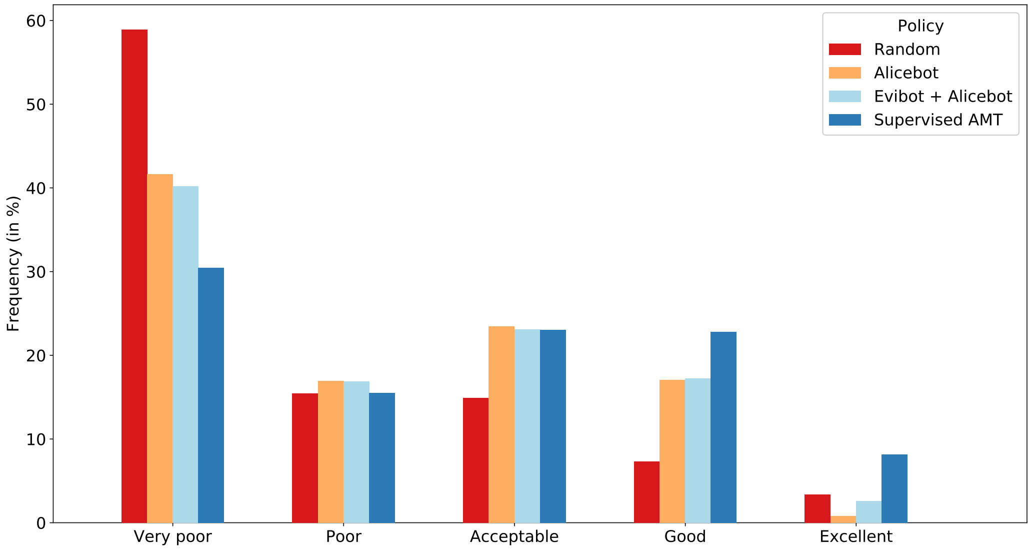


Figure 6: Amazon Mechanical Turk class frequencies on the test set w.r.t. different policies.

Table 2 shows the performance w.r.t. Pearson correlation coefficient, Spearman's rank correlation coefficient and mean squared error. The metrics are computed after linearly transforming the AMT class categories to the scalar output score (i.e. by taking the dot-product between the one-hot class vector and the vector $[1.0, 2.0, 3.0, 4.0, 5.0]$). The *Average Predictor* is a baseline model, which always predicts with the average output score. As shown, *Supervised AMT* achieves a Pearson correlation coefficient of 0.40, a Spearman's rank correlation coefficient of 0.38 and a significant reduction in mean squared error. This indicates *Supervised AMT* performs significantly better than the baseline.

Figure 6 shows the performance w.r.t. each AMT label class. In addition to *Supervised AMT*, the figure shows the performance of three baseline policies: 1) *Random*, which selects a response at random, 2) *Alicebot*, which selects an *Alicebot* response if available and otherwise selects a response at random, and 3) *Evibot + Alicebot*, which selects an *Evibot* response if available and otherwise selects an *Alicebot* response. For each policy, the figure shows the percentage of responses selected by the policy belonging to a particular AMT label class. In one end of the spectrum, we observe that *Supervised AMT* has a ~30% point reduction compared to *Random* in responses belonging to the *"very poor"* class. For the same AMT label class, *Supervised AMT* has a reduction of ~10% points compared to *Alicebot* and *Evibot + Alicebot*. In the other end of the spectrum, we observe that *Supervised AMT* performs significantly better than the three baselines w.r.t. the classes *"good"* and *"excellent"*. In particular, *Supervised AMT* reaches ~8% responses belonging to the class *"excellent"*. This is more than double compared to all three baseline policies. This demonstrates that *Supervised AMT* has learned to select *"good"* and *"excellent"* responses, while avoiding *"very poor"* and *"poor"* responses.

Overall, the results show that *Supervised AMT* improves substantially over all baseline policies. Nevertheless, ~46% of the *Supervised AMT* responses belong to the classes *"very poor"* and *"poor"*. This implies that there is ample space for improving both *Supervised AMT* and the set of candidate responses (i.e. the system's response models).

4.4 Supervised Learned Reward: Learning with a Learned Reward Function

In the first scoring model *Supervised AMT* we fixed the last output layer weights to $[1.0, 2.0, 3.0, 4.0, 5.0]$. In other words, we assigned a score of 1.0 for *very poor* responses, 2.0 for *poor* responses, 3.0 for *acceptable* responses, and so on. It's not clear whether this score is correlated with scores given by real-world Alexa users, which is what we ultimately want to optimize the system for. This section describes another approach, which remedies this problem by learning to predict the Alexa user scores based on previously recorded dialogues.

Learned Reward Function: Let h_t be a dialogue history and let a_t be the corresponding response, given by the system at time t. We aim to learn a linear regression model, g_ϕ, which predicts the corresponding return (Alexa user score) at the current dialogue turn:

$$g_\phi(h_t, a_t) \in [1, 5], \tag{7}$$

where ϕ are the model parameters. We call this a *reward model*, since it directly models the Alexa user score, which we aim to maximize.

Let $\{h_t^d, a_t^d, R^d\}_{d,t}$ be a set of examples, where t denotes the time step and d denotes the dialogue. Let $R^d \in [1, 5]$ denote the observed real-valued return for dialogue d.

Specifically, we set R^d to be the Alexa user score given at the end of dialogue d. It's optional for users to a give a score; users are prompted to give a score at the end, but they may opt out by stopping the application. Although not all users give scores, we do not consider examples without scores.[19] Furthermore, users are encouraged to give a score in the range $1 - 5$. The majority of users give whole number (integer) scores, but some users give decimal scores (e.g. 3.5). Therefore, we treat R^d as a real-valued number in the range $1 - 5$.

We learn ϕ by minimizing the squared error between the model's prediction and the observed return:

$$\hat{\phi} = \arg\max_{\phi} \sum_d \sum_t (g_\phi(h_t^d, a_t^d) - R^d)^2 \tag{8}$$

As before, we optimize the model parameters with mini-batch stochastic gradient descent (SGD) using Adam. We use L2 regularization with coefficients in the set $\{10.0, 1.0, 0.1, 0.01, 0.001, 0.0001, 0.00001, 0.0\}$. We select the coefficient with the smallest squared error on a hold-out dataset.

As input to the reward model we compute 23 features based on the dialogue history and a candidate response. As training data is scarce, we use only higher-level features:

AMT label class: A vector indicating the probability of the AMT label classes for the candidate response, computed using *Supervised AMT*, as well as the probability that the candidate response has priority. If the candidate response has priority, the vector is zero in all entries, except the last entry corresponding to the priority class: $[0.0, 0.0, 0.0, 0.0, 0.0, 1.0]$.

Generic response: A binary feature, which is 1.0 when the response only contains stop-words and otherwise zero.

Response length: The number of words in the response, and the square root of the number of words in the response.

Dialogue act: A one-hot vector, indicating whether the last user utterance's dialogue is a *request*, a *question*, a *statement* or contains profanity (Stolcke et al. 2000).

[19] By ignoring dialogues without Alexa user scores, we introduce a significant bias in our reward model. In particular, it seems likely that the users who did not provide a score either found the system to be very poor or to lack particular functions/features they expected (e.g. *non-conversational activities*, such as playing games or taking quizzes.). A related problem arises in medical statistics, when patients undergo a treatment and, later, their outcome is not observed.

Sentiment class: A one-hot vector, indicating whether the last user utterance's dialogue is negative, neutral or positive.

Generic user utterance: A binary feature, which is 1.0 when the last user utterance only contains stop-words, and otherwise zero.

User utterance length: The number of words in the last user utterance, and the square root of the number of words in the response.

Confusion indicator: A binary feature, which is 1.0 when the last user utterance is very short (less than three words) and contains at least one word indicating the user is confused (e.g. *"what"*, *"silly"*, *"stupid"*).

Dialogue length: The number of dialogue turns so far, as well as the square root and logarithm of the number of dialogue turns.

In total, our dataset for training the reward model has 4340 dialogues. We split this into a training set with 3255 examples and a test set with 1085 examples.

To increase data efficiency, we learn an ensemble model through a variant of the bagging technique (Breiman 1996). We create 5 new training sets, which are shuffled versions of the original training set. Each shuffled dataset is split into a sub-training set and sub-hold-out set. The sub-hold-out sets are created such that the examples in one set do not overlap with other sub-hold-out sets. A reward model is trained on each sub-training set, with its hyper-parameters selected on the sub-hold-out set. This increases data efficiency by allowing us to re-use the sub-hold-out sets for training, which would otherwise not have been used. The final reward model is an ensemble, where the output is an average of the underlying linear regression models.

The reward model obtains a mean squared error of 0.96 and a Spearman's rank correlation coefficient of 0.19 w.r.t. the real Alexa user on the test set. In comparison, a model predicting with the average user score obtains a mean squared error of 0.99 and (because it outputs a constant) a Spearman's rank correlation coefficient of zero. Although the reward model is better than predicting the average, its correlation is relatively low. There are two reasons for this. First, the amount of training data is very small. This makes it difficult to learn the relationships between the features and the Alexa user scores. Second, the Alexa user scores are likely to have high variance because, they are influenced by many different factors. The score of the user may be determined by a single turn in the dialogue (e.g. a single misunderstanding at the end of the dialogue could result in a very low user score, even if all the previous turns in the dialogue were excellent). The score of the user may be affected by the accuracy of the speech recognition module. More speech recognition errors will inevitably lead to frustrated users. In a preliminary study, we found that Spearman's rank correlation coefficient between the speech recognition confidences and the Alexa user scores was between $0.05 - 0.09$. In comparison to correlations with other factors, this implies that speech recognition performance plays an important role in determining user satisfaction.[20] In addition, extrinsic factors are likely to have a substantial influence on the user scores. The user scores are likely to depend not only on the dialogue, but also on the user's profile (e.g. whether the user is an adult or a child), the environment (e.g. whether the user is alone with the system or several users are taking turns conversing with the system), the user's expectations towards the system before starting the conversation (e.g. whether the system is capable of playing games) and the emotional state of the user (e.g. the user's mood).

Training: To prevent overfitting, we do not train the scoring model from scratch with the reward model as target. Instead, we first initialize the scoring model with the parameters of the *Supervised AMT*, and then train it with the reward model outputs to minimize the squared error:

$$\hat{\theta} = \arg\max_{\theta} \sum_{d} \sum_{t} (f_{\theta}(h_t^d, a_t^d) - g_{\phi}(h_t^d, a_t^d))^2, \tag{9}$$

As before, we optimize the model parameters with stochastic gradient descent using Adam. As training this model does not depend on AMT labels, training is carried out on recorded dialogues. We train on several thousand recorded dialogue examples, where about 80% are used for training and about 20% are used as hold-out set. No regularization is used. We early stop on the squared error of the hold-out dataset w.r.t. Alexa user scores predicted by the reward model. As this scoring model was trained with a learned reward function, we call it *Supervised Learned Reward*.

[20] This was confirmed by manual inspection of the conversation logs, where the majority of conversations had several speech recognition errors. In conversations with an excessive number of speech recognition errors (perhaps due to noisy environments), the users' utterances clearly showed frustration with the system.

4.5 Off-policy REINFORCE

As discussed earlier, one way to parametrize the policy is as a discrete probability distribution over actions. This parametrization allows us to learn the policy directly from recorded dialogues through a set of methods known as *policy gradient* methods. This section describes one such approach.

Off-policy Reinforcement Learning: We use a variant of the classical *REINFORCE* algorithm (Williams 1992, Precup 2000, Precup et al. 2001), which we call *Off-policy REINFORCE*. Recall eq. (4), where the policy's distribution over actions is parametrized as softmax function applied to a function f_θ with parameters θ. As before, let $\{h_t^d, a_t^d, R^d\}_{d,t}$ be a set of examples, where h_t^d is the dialogue history for dialogue d at time t, a_t^d is the agent's action for dialogue d at time t and R^d is the return for dialogue d. Let D be the number of dialogues and let T^d be the number of turns in dialogue d. Further, let θ_d be the parameters of the stochastic policy π_{θ_t} used during dialogue d. The *Off-policy REINFORCE* algorithm updates the policy parameters θ by:

$$\Delta\theta \propto c_t^d \; \nabla_\theta \log \pi_\theta(a_t^d | h_t^d) \; R^d \quad \text{where } d \sim \text{Uniform}(1, D) \text{ and } t \sim \text{Uniform}(1, T^d), \tag{10}$$

where c_t^d is the importance weight ratio:

$$c_t^d \overset{\text{def}}{=} \frac{\prod_{t'=1}^{t} \pi_\theta(a_{t'}^d | h_{t'}^d)}{\prod_{t'=1}^{t} \pi_{\theta_d}(a_{t'}^d | h_{t'}^d)}. \tag{11}$$

This ratio corrects for the discrepancy between the learned policy π_θ and the policy under which the data was collected π_{θ_t} (sometimes referred to as the behaviour policy). It up-weights examples with high probability under the learned policy and down-weights examples with low probability under the learned reward function.

The intuition behind the algorithm can be illustrated by analogy with learning from trial and error. When an example has a high return (i.e. high user score), the term $\nabla_\theta \log \pi_\theta(a_t^d | h_t^d) \; R^d$ will be a vector pointing in a direction increasing the probability of taking action a_t^d. On the other hand, when an example has low return (i.e. low user score), the term $\nabla_\theta \log \pi_\theta(a_t^d | h_t^d) \; R^d$ will be a vector close to zero or a vector pointing in the opposite direction, hence decreasing the probability of taking action a_t^d.

The importance ratio c_t^d is known to exhibit very high, possibly infinite, variance (Precup et al. 2001). Therefore, we truncate the products in the nominator and denominator to only include the current time step t:

$$c_{t,\text{trunc.}}^d \overset{\text{def}}{=} \frac{\pi_\theta(a_t^d | h_t^d)}{\pi_{\theta_d}(a_t^d | h_t^d)}. \tag{12}$$

This induces bias in the learning process, but also acts as a regularizer.

Reward Shaping: As mentioned before, one problem with the *Off-policy REINFORCE* algorithm presented in eq. (10) is that it suffers from high variance (Precup et al. 2001). The algorithm uses the return, observed only at the very end of an episode, to update the policy's action probabilities for all intermediate actions in an episode. With a small number of examples, the variance in the gradient estimator is overwhelming and this could easily lead the agent to over-estimate the utility of poor actions and, vice versa, to under-estimate the utility of good actions. One remedy for this problem is *reward shaping*, where the reward at each time step is estimated using an auxiliary function (Ng et al. 1999). For our purpose, we propose a simple variant of reward shaping which takes into account the sentiment of the user. When the user responds with a negative sentiment (e.g. an angry comment), we will assume that the preceding action was highly inappropriate and assign it a reward of zero. Given a dialogue d, at each time t we assign reward r_t^d:

$$r_t^d \overset{\text{def}}{=} \begin{cases} 0 & \text{if user utterance at time } t+1 \text{ has negative sentiment,} \\ \dfrac{R^d}{T^d} & \text{otherwise.} \end{cases} \tag{13}$$

With reward shaping and truncated importance weights, the learning update becomes:

$$\Delta\theta \propto c_{t,\text{trunc.}}^d \nabla_\theta \log \pi_\theta(a_t^d | h_t^d) \; r_t^d \quad \text{where } d \sim \text{Uniform}(1, D), t \sim \text{Uniform}(1, T^d), \tag{14}$$

Off-policy Evaluation: To evaluate the policy, we estimate the expected return (Precup 2000):

$$\mathrm{R}_{\pi_\theta}[R] \ \lessapprox \ \sum_{d,t} c^d_{t,\text{trunc.}} \, r^d_t. \tag{15}$$

Furthermore, by substituting r^d_t with a constant reward of 1.0 for each time step, we can compute the estimated number of time steps per episode under the policy. As will be discussed later, this is an orthogonal metric based on which we can analyse and evaluate each policy. However, this estimate does not include the number of priority responses, since there are no actions for the agent to take when there is a priority response.

Training: We initialize the policy model with the parameters of *Supervised AMT*, and then train the parameters w.r.t. eq. (14) with stochastic gradient descent using Adam. We use a set of a few thousand dialogues recorded between Alexa users and a preliminary version of the system. About 60% of these examples are used for training, and about 20% are used for development and testing. To reduce the risk of overfitting, we only train the weights related to the second last layer using *off-policy REINFORCE*. We use a random grid search with different hyper-parameters, which include the temperature parameter λ and the learning rate. We select the hyper-parameters with the highest expected return on the development set.

4.6 Off-policy REINFORCE with Learned Reward Function

Similar to the *Supervised Learned Reward* policy, we may use the reward model for training with the *Off-policy REINFORCE* algorithm. This section describes how we combine the two approaches.

Reward Shaping with Learned Reward Model: We use the reward model to compute a new estimate for the reward at each time step in each dialogue:

$$r^d_t \overset{\text{def}}{=} \begin{cases} 0 & \text{if user utterance at time } t+1 \text{ has negative sentiment,} \\ g_\phi(h_t, a_t) & \text{otherwise.} \end{cases} \tag{16}$$

This is substituted into eq. (14) for training and into eq. (15) for evaluation.

Training: As with *Off-policy REINFORCE*, we initialize the policy model with the parameters of the *Supervised AMT* model, and then train the parameters w.r.t. eq. (14) with mini-batch stochastic gradient descent using Adam. We use the same set of dialogues and split as *Off-policy REINFORCE*. We use a random grid search with different hyper-parameters, As before, to reduce the risk of overfitting, we only train the weights related to the second last layer using this method. which include the temperature parameter λ and the learning rate, and select the hyper-parameters with the highest expected return on the development set. In this case, the expected return is computed according to the learned reward model. As this policy uses the learned reward model, we call it *Off-policy REINFORCE Learned Reward*.

4.7 Q-learning with the Abstract Discourse Markov Decision Process

The approaches described so far have each their own advantages and disadvantages. One way to quantify their differences is through a decomposition known as the *bias-variance trade-off*. At one end of the spectrum, the *Supervised AMT* policy has low variance, because it was trained with hundreds of thousands of human annotations at the level of each model response. However, for the same reason, *Supervised AMT* incurs a substantial bias, because the human annotations do not reflect the real user satisfaction for an entire conversation. At the other end of the spectrum, *Off-policy REINFORCE* suffers from high variance, because it was trained with only a few thousand dialogues and corresponding user scores. To make matters worse, the user scores are affected by many external factors (e.g. user profile, user expectations, and so on) and occur at the granularity of an entire conversation. Nevertheless, this method incurs low bias because it directly optimizes the objective metric we care about (i.e. the user score).[21] By utilizing a learned reward function, *Supervised Learned Reward* and *Off-policy REINFORCE Learned Reward* suffer less from bias, but since the learned reward function has its own variance component, they are both bound to have higher variance. In general, finding the optimal trade-off between bias and variance can be notoriously difficult. In

[21] Due to truncated importance weights, however, the *off-policy REINFORCE* training procedure is still biased.

this section we propose a novel method for trading off bias and variance by learning the policy from simulations in an approximate Markov decision process.

Motivation A Markov decision process (MDP) is a framework for modeling sequential decision making (Sutton & Barto 1998). In the general setting, an MDP is a model consisting of a discrete set of states H, a discrete set of actions A, a transition distribution function P, a reward distribution function R, and a discount factor γ. As before, an agent aims to maximize its reward during each episode. Let t denote the time step of an episode with length T. At time step t, the agent is in state $h_t \in H$ and takes action $a_t \in A$. Afterwards, the agent receives reward $r_t \sim R(h_t, a_t)$ and transitions to a new state $h_{t+1} \sim P(h_t|a_t)$.

Given an MDP model for open-domain conversations, there are dozens of algorithms we could apply to learn the agent's policy (Sutton & Barto 1998). Unfortunately, such an MDP is difficult to build or estimate. We could try to naively estimate one from the recorded dialogues, but this would require solving two extremely difficult problems. First, we would need to learn the transition distribution P, which outputs the next user utterance in the dialogue given the dialogue history. This problem is likely to be as difficult as our original problem of finding an appropriate response to the user! Second, we would need to learn the reward distribution R for each time step. However, as we have shown earlier, it is very difficult to learn to predict the user score for an entire dialogue. Given the data we have available, estimating the reward for a single turn is likely also going to be difficult. Instead, we propose to tackle the problem by splitting it into three smaller parts.

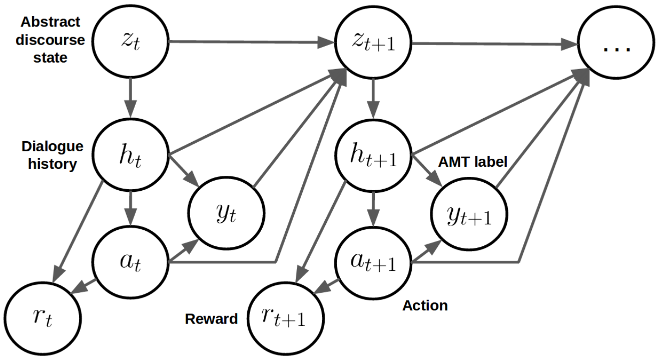


Figure 7: Probabilistic directed graphical model for the *Abstract Discourse Markov Decision Process*. For each time step t, z_t is a discrete random variable which represents the abstract state of the dialogue, h_t represents the dialogue history, a_t represents the action taken by the system (i.e. the selected response), y_t represents the sampled AMT label and r_t represents the sampled reward.

The Abstract Discourse Markov Decision Process The model we propose to learn is called the *Abstract Discourse MDP*. As illustrated in Figure 7, the model follows a hierarchical structure at each time step. At time t, the agent is in state $z_t \in Z$, a discrete random variable representing the *abstract discourse state*. This variable only represents a few high-level properties related to the dialogue history. We define the set Z is the Cartesian product:

$$Z = Z_{\text{Dialogue act}} \times Z_{\text{User sentiment}} \times Z_{\text{Generic user utterance}}, \tag{17}$$

where $Z_{\text{Dialogue act}}$, $Z_{\text{User sentiment}}$ and $Z_{\text{Generic user utterance}}$ are three discrete sets. The first set consists of 10 dialogue acts: $Z_{\text{Dialogue act}} = \{\text{Accept}, \text{Reject}, \text{Request}, \text{Politics}, \text{Generic Question}, \text{Personal Question}, \text{Statement}, \text{Greeting}, \text{Goodbye}, \text{Other}\}$. These dialogue acts represent the high-level intention of the user's utterance (Stolcke et al. 2000). The second set consists of sentiments types: $Z_{\text{User sentiment}} = \{\text{Negative}, \text{Neutral}, \text{Positive}\}$. The third set represent a binary variable: $Z_{\text{Generic user utterance}} = \{\text{True}, \text{False}\}$. This variable is *True* only when the user utterance is generic and topic-independent (i.e. when the user utterance only contains stop-words). We build a hand-crafted deterministic classifier, which maps a dialogue history to the corresponding classes in $Z_{\text{Dialogue act}}$, $Z_{\text{User sentiment}}$ and $Z_{\text{Generic user utterance}}$. We denote this mapping $f_{h \to z}$. Although we only consider dialogue acts, sentiment and generic utterances, it is trivial to expand the *abstract discourse state* with other types of discrete or real-valued variables.

Given a sample z_t, the *Abstract Discourse MDP* samples a dialogue history h_t from a finite set of dialogue histories H. In particular, h_t is sampled at uniformly random from the set of dialogue histories where the last utterance is mapped to z_t:

$$h_t \sim P(h|H, f_{h \to z}, z_t) \overset{\text{def}}{=} \text{Uniform}(\{h \mid h \in H \text{ and } f_{h \to z}(h) = z_t\}). \tag{18}$$

In other words, h_t is a dialogue history where dialogue act, user sentiment and generic property is identical to the discrete variable z_t.

For our purpose, H is the set of all recorded dialogues between Alexa users and a preliminary version of the system. This formally makes the *Abstract Discourse MDP* a *non-parametric* model, since sampling from the model requires access to the set of recorded dialogue histories H. This set grows over time when the system is deployed in practice. This is useful, because it allows to continuously improve the policy as new data becomes available. Further, it should be noted that the set Z is small enough that every possible state is observed several times in the recorded dialogues.

Given a sample h_t, the agent chooses an action a_t according to its policy $\pi_\theta(a_t|h_t)$, with parameters θ. A reward r_t is then sampled such that $r_t \sim R(h_t, a_t)$, where R is a distribution function. In our case, we use the probability function $P_{\hat{\theta}}$, where the parameters $\hat{\theta}$ are estimated using supervised learning on AMT labels in eq. (6). We specify a reward of -2.0 for a *"very poor"* response class, a reward of -1.0 for a *"poor"* response class, a reward of 0.0 for an *"acceptable"* response class, a reward of 1.0 for a *"good"* response class and a reward of 2.0 for an *"excellent"* response class. To reduce the number of hyperparameters, we use the expected reward instead of a sample:[22]

$$r_t = P_{\hat{\theta}}(y|h_t, a_t)^{\text{T}}[-2.0, -1.0, 0.0, 1.0, 2.0]. \tag{19}$$

Next, a variable $y_t \in \{$"very poor", "poor", "acceptable", "good", "excellent"$\}$ is sampled:

$$y_t \sim P_{\hat{\theta}}(y|h_t, a_t). \tag{20}$$

This variable represents one *appropriateness interpretation* of the output. This variable helps predict the future state z_{t+1}, because the overall appropriatness of a response has a significant impact on the user's next utterance (e.g. very poor responses often cause users to respond with *What?* or *I don't understand.*).

Finally, a new state z_{t+1} is sampled according to $P_{\hat{\psi}}$:

$$z_{t+1} \sim P_{\hat{\psi}}(z|z_t, h_t, a_t, y_t). \tag{21}$$

where $P_{\hat{\psi}}$ is the transition distribution with parameters $\hat{\psi}$. The transition distribution is parametrized by three independent two-layer MLP models, which take as input the same features as the scoring function, as well as 1) a one-hot vector representing the sampled response class y_t, 2) a one-hot vector representing the dialogue act of the last user utterance, 3) a one-hot vector representing the sentiment of the last user utterance, 4) a binary variable indicating whether the last user utterance was generic, and 5) a binary variable indicating whether the last user utterance contained a wh-word (e.g. *what*, *who*). The first MLP predicts the next dialogue act, the second MLP predicts the next sentiment type and the third MLP predicts whether the next user utterance is generic. The dataset for training the MLPs consists of $499,757$ transitions, of which 70% are used for training and 30% for evaluation. The MLPs are trained with maximum log-likelihood using mini-batch stochastic gradient descent. We use Adam and early-stop on a hold-out set. Due to the large number of examples, no regularization is used. The three MLP models obtain a joint perplexity of 19.51. In comparison, a baseline model, which always assigns the average class frequency as the output probability obtains a perplexity of 23.87. On average, this means that roughly $3-4$ possible z_{t+1} states can be eliminated by conditioning on the previous variables z_t, h_t, a_t and y_t. In other words, the previous state z_t and h_t, together with the agent's action a_t has a significant effect on the future state z_{t+1}. This means that an agent trained in the *Abstract Discourse MDP* has the potential to learn to take into account future states of the dialogue when selecting its action. This is in contrast to policies learned using supervised learning, which do not consider future dialogue states.

The idea of modeling a high-level abstraction of the dialogue, z_t, is related to the dialogue state tracking challenge (Williams et al. 2013, 2016). In this challenge, the task is to map the dialogue

[22] For example, if we were to use a Gaussian distribution, we would have to at least also specify the variance parameter.

history to a discrete state representing all salient information about the dialogue. Unlike the dialogue state tracking challenge, however, the variable z_t only includes limited salient information about the dialogue. For example, in our implementation, z_t does not include topical information. As such, z_t is only a partial representation of the dialogue history.

Training Given the *Abstract Discourse MDP*, we are now able to learn policies directly from simulations. We use *Q-learning* with *experience replay* to learn the policy, since it is simple and has been shown to be effective with policies parametrized by neural networks (Mnih et al. 2013, Lin 1993). For experience replay, we use a memory buffer of size 1000. We use an ϵ-greedy exploration scheme with $\epsilon = 0.1$. We experiment with discount factors $\gamma \in \{0.1, 0.2, 0.5\}$. As before, the parameters are updated using Adam. To reduce the risk of overfitting, we only train the weights related to the final output layer and the skip-connection (shown in dotted lines in Figure 2) using Q-learning.

Training is carried out in two alternating phases. We train the policy for 100 episodes. Then, we evaluate the policy for 100 episodes w.r.t. average return. Afterwards, we continue training the policy for another 100 episodes. During evaluation, each dialogue history is sampled from a separate set of dialogue histories, H_{Eval}, which is disjoint from the set of dialogue histories, H_{Train} used at training time. This ensures that the policy is not *overfitting* our finite set of dialogue histories. For each hyper-parameter combination, we train the policy between 400 and 600 episodes. We select the policy which performs best w.r.t. average return. To keep notation brief, we call this policy *Q-learning AMT*.

4.8 Preliminary Evaluation

In this section, we carry out a preliminary evaluation of the response model selection policies.

Table 3: Off-policy evaluation w.r.t. expected (average) Alexa user score and number of time steps (excluding priority responses) on test set.

Policy	**Alexa user score**	**Time steps**
Supervised AMT	2.06	8.19
Supervised Learned Reward	0.94	3.66
Off-policy REINFORCE	**2.45**	**10.08**
Off-policy REINFORCE Learned Reward	1.29	5.02
Q-learning AMT	2.08	8.28

Off-policy Evaluation: One way to evaluate the selection policies is by using the off-policy evaluation given in eq. (15). This equation provides an estimate of the expected Alexa user score under each policy.[23] As described earlier, the same equation can be used to estimate the expected number of time steps per episode (excluding priority responses).

The expected (average) Alexa user score and number of time steps per episode (excluding priority responses) are given in Table 3. Here we observe that the *Off-policy REINFORCE* performs best followed by *Q-learning AMT* and *Supervised AMT* w.r.t. expected Alexa user score. *Off-policy REINFORCE* reaches 2.45, which is a major 17.8% improvement over the second best performing model *Q-learning AMT*. However, this advantage should be taken with a grain of salt. As discussed earlier, the off-policy evaluation in eq. (15) is a biased estimator since the importance weights have been truncated. Moreover, *Off-policy REINFORCE* has been trained specifically to maximize this biased estimator, while all other policies have been trained to maximize other objective functions. Similarly, w.r.t. expected number of time steps, *Off-policy REINFORCE* reaches the highest number of time steps followed by *Q-learning AMT* and *Supervised AMT*. As before, we should take this result with a grain of salt, since this evaluation is also biased and does not take into account priority responses. Further, it's not clear that increasing the number of time steps will increase user scores. Nevertheless, *Off-policy REINFORCE*, *Q-learning AMT* and *Supervised AMT* appear to be our prime candidates for further experiments.

Response Model Selection Frequency: Figure 8 shows the frequency with which *Supervised AMT*, *Off-policy REINFORCE* and *Q-learning AMT* select different response models. We observe that the

[23] For the policies parametrized as action-value functions, we transform eq. (2) to eq. (4) by setting $f_\theta = Q_\theta$ and fitting the temperature parameter λ on the *Off-policy REINFORCE* development set.

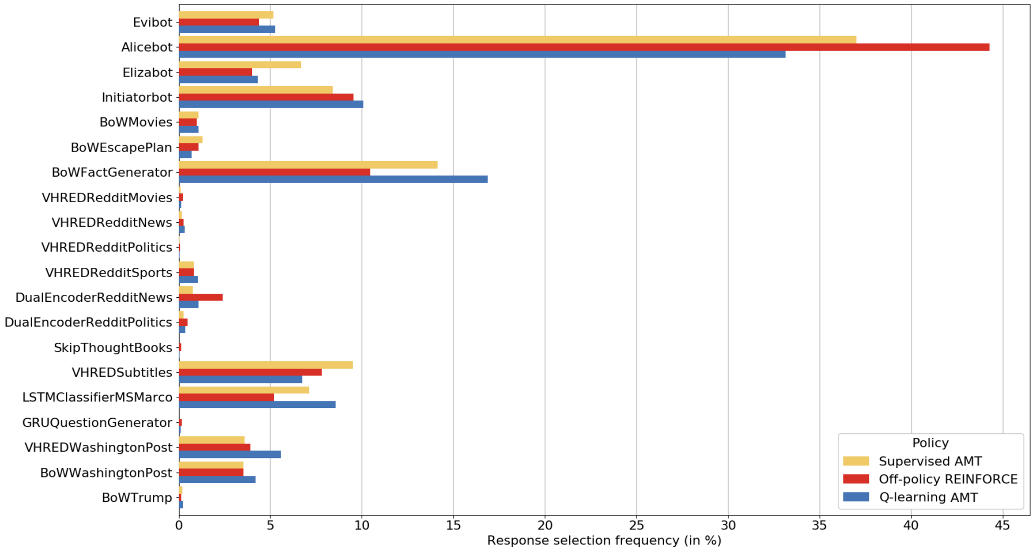


Figure 8: Response model selection probabilities across response models for *Supervised AMT*, *Off-policy REINFORCE* and *Q-learning AMT* on the AMT label test dataset.

policy learned using *Off-policy REINFORCE* tends to strongly prefer *Alicebot* responses over other models. The *Alicebot* responses are among the safest and most topic-dependent, generic responses in the system, which suggests that *Off-policy REINFORCE* has learned a highly *risk averse strategy*. On the other hand, the *Q-learning AMT* policy selects *Alicebot* responses substantially less often than both *Off-policy REINFORCE* and *Supervised AMT*. Instead, *Q-learning AMT* tends to prefer responses retrieved from Washington Post and from Google search results. These responses are semantically richer and have the potential to engage the user more deeply in a particular topic, but they are also more risky (e.g. a bad choice could derail the entire conversation.). This suggests that *Q-learning AMT* has learned a more *risk tolerant strategy*. One possible explanation for this difference is that *Q-learning AMT* was trained using simulations. By learning online from simulations, the policy has been able to explore new actions and discover high-level strategies lasting multiple time steps. In particular, the policy has been allowed to experiment with riskier actions and to learn *remediation* or *fall-back strategies*, in order to handle cases where a risky action fails. This might also explain its stronger preference for *BoWFactGenerator* responses, which might be serving as a fall-back strategy by outputting factual statements on the current topic. This would have been difficult to learn for *Off-policy REINFORCE*, since the sequence of actions for such high-level strategies are sparsely observed in the data and, when they are observed, the corresponding returns (Alexa user scores) have high variance.

A second observation is that *Q-learning AMT* has the strongest preference for *Initiatorbot* among the three policies. This could indicate that *Q-learning AMT* leans towards a *system-initiative strategy* (e.g. a strategy where the system tries to maintain control of the conversation by asking questions, changing topics and so on). Further analysis is needed to confirm this.

Abstract Discourse MDP Evaluation Next, we can evaluate the performance of each policy w.r.t. simulations in the *Abstract Discourse MDP*. We simulate 500 episodes under each policy and evaluate it w.r.t. average return, average reward per time step and dialogue length. In addition to evaluating the five policies described earlier, we also evaluate three heuristic policies: 1) a policy selecting responses at random called *Random*, 2) a policy selecting only *Alicebot* responses called *Alicebot*,[24] and 3) a policy selecting *Evibot* responses when possible and *Alicebot* responses otherwise, called *Evibot + Alicebot*[25]. Evaluating these models will serve to validate the approximate MDP.

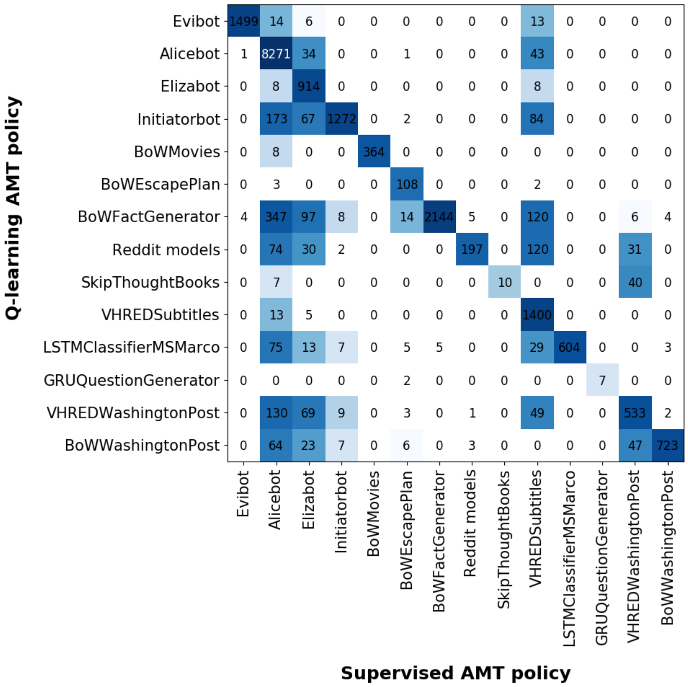
[24] When there are no valid *Alicebot* responses, this policy selects a response at random.

[25] When there are no valid *Evibot* or *Alicebot* responses, this policy selects a response at random.

Table 4: Policy evaluation using the *Abstract Discourse MDP* w.r.t. average return, average reward per time step and average episode length on dev set (± standard deviations). The reward function is based on *Supervised AMT*.

Policy	**Average return**	**Average reward per time step**	**Average dialogue length**
Random	-32.18 ± 31.77	-0.87 ± 0.24	34.29 ± 33.02
Alicebot	-15.56 ± 15.61	-0.37 ± 0.16	42.01 ± 42.00
Evibot + Alicebot	-11.33 ± 12.43	-0.29 ± 0.19	37.5 ± 38.69
Supervised AMT	$\mathbf{-6.46 \pm 8.01}$	$\mathbf{-0.15 \pm 0.16}$	$\mathbf{42.84 \pm 42.92}$
Supervised Learned Reward	-24.19 ± 23.30	-0.73 ± 0.27	31.91 ± 30.09
Off-policy REINFORCE	$\mathbf{-7.30 \pm 8.90}$	$\mathbf{-0.16 \pm 0.16}$	$\mathbf{43.24 \pm 43.58}$
Off-policy REINFORCE Learned Reward	-10.19 ± 11.15	-0.28 ± 0.19	35.51 ± 35.05
Q-learning AMT	$\mathbf{-6.54 \pm 8.02}$	$\mathbf{-0.15 \pm 0.18}$	$\mathbf{40.68 \pm 39.13}$

The results are given in Table 4. We observe that *Supervised AMT* performs best w.r.t. average return and average reward per time step. However, this comes as no surprise. The reward function in the MDP is defined as *Supervised AMT*, so by construction this policy achieves the highest reward per time step. Next we observe that *Q-learning AMT* is on par with *Supervised AMT*, both achieving same -0.15 average reward per time step. Second in line comes *Off-policy REINFORCE*, achieving an average reward per time step of -0.16. However, *Off-policy REINFORCE* also achieved the highest average dialogue length of 43.24. At the other end of the spectrum comes, as expected, the *Random* policy performing worst w.r.t. all metrics. In comparison, both *Alicebot* and *Evibot + Alicebot* perform better w.r.t. all metrics, with *Evibot + Alicebot* achieving the best average return and average reward per time step out of the three heuristic policies. This validates the utility of the *Abstract Discourse MDP* as an environment for training and evaluating policies. Overall, *Off-policy REINFORCE*, *Q-learning AMT* and *Supervised AMT* still appear to be the best performing models in the preliminary evaluation.


Q-learning AMT policy
Supervised AMT policy

	Evibot	Alicebot	Elizabot	Initiatorbot	BoWMovies	BoWEscapePlan	BoWFactGenerator	Reddit models	SkipThoughtBooks	VHREDSubtitles	LSTMClassifierMSMarco	GRUQuestionGenerator	VHREDWashingtonPost	BoWWashingtonPost
Evibot	1499	14	6	0	0	0	0	0	0	13	0	0	0	0
Alicebot	1	8271	34	0	0	1	0	0	0	43	0	0	0	0
Elizabot	0	8	914	0	0	0	0	0	0	8	0	0	0	0
Initiatorbot	0	173	67	1272	0	2	0	0	0	84	0	0	0	0
BoWMovies	0	8	0	0	364	0	0	0	0	0	0	0	0	0
BoWEscapePlan	0	3	0	0	0	108	0	0	0	2	0	0	0	0
BoWFactGenerator	4	347	97	8	0	14	2144	5	0	120	0	0	6	4
Reddit models	0	74	30	2	0	0	0	197	0	120	0	0	31	0
SkipThoughtBooks	0	7	0	0	0	0	0	0	10	0	0	0	40	0
VHREDSubtitles	0	13	5	0	0	0	0	0	0	1400	0	0	0	0
LSTMClassifierMSMarco	0	75	13	7	0	5	5	0	0	29	604	0	0	3
GRUQuestionGenerator	0	0	0	0	0	2	0	0	0	0	0	7	0	0
VHREDWashingtonPost	0	130	69	9	0	3	0	1	0	49	0	0	533	2
BoWWashingtonPost	0	64	23	7	0	6	0	3	0	0	0	0	47	723

Figure 9: Contingency table comparing selected response models between *Supervised AMT* and *Q-learning AMT*. The cells in the matrix show the number of times the *Supervised AMT* policy selected the row response model and the *Q-learning AMT* policy selected the column response model. The cell frequencies were computed by simulating 500 episodes under the Q-learning policy in the *Abstract Discourse MDP*. Note that all models retrieving responses from Reddit have been agglomerated into the class *Reddit models*.

Finally, we compare *Q-learning AMT* with *Supervised AMT* w.r.t. the action taken in states from episodes simulated in the *Abstract Discourse MDP*. As shown in Figure 9, the two policies diverge w.r.t. several response models. When *Supervised AMT* would have selected topic-independent, generic *Alicebot* and *Elizabot* responses, *Q-learning AMT* often selects *BoWFactGenerator*, *Initiatorbot* and *VHREDWashingtonPost* responses. For example, there were 347 instances where *Supervised AMT* selected *Alicebot*, but where *Q-learning AMT* selected *BoWFactGenerator*. Similarly, where *Supervised AMT* would have preferred generic *VHREDSubtitle* responses, *Q-learning AMT* often selects responses from *BoWFactGenerator*, *InitiatorBot* and *VHREDRedditSports*. This supports our previous analysis showing that *Q-learning AMT* has learned a more risk tolerant strategy, which involves response models with semantically richer content.

In the next section, we evaluate these policies with real-world users.

5 A/B Testing Experiments

To evaluate the dialogue manager policies described in the previous section, we carry out A/B testing experiments.

During each A/B testing experiment, we evaluate several policies for selecting the response model. When Alexa users start a conversation with the system, they are automatically assigned to a random policy and afterwards their dialogues and final scores are recorded. However, the distribution over Alexa users is bound to change over time. Different types of users will use the system depending on the time of day, weekday and holiday season. In addition, the user expectations towards our system change over time as they interact with other socialbots in the competition. In other words, we must consider the Alexa user distribution as following a non-stationary stochastic process. Therefore,

Table 5: First A/B testing experiment with six different policies (The $\pm$ 95% confidence intervals). Star * indicates policy is significantly better than other policies at 95% statistical significance level.

Policy	**User score**	**Dialogue length**	**Pos. utterances**	**Neg. utterances**
Evibot + Alicebot	2.86 ± 0.22	31.84 ± 6.02	$2.80\% \pm 0.79$	$5.63\% \pm 1.27$
Supervised AMT	2.80 ± 0.21	34.94 ± 8.07	$\mathbf{4.00\% \pm 1.05}$	$8.06\% \pm 1.38$
Supervised Learned Reward	2.74 ± 0.21	27.83 ± 5.05	$2.56\% \pm 0.70$	$6.46\% \pm 1.29$
Off-policy REINFORCE	2.86 ± 0.21	$\mathbf{37.51 \pm 7.21}$	$3.98\% \pm 0.80$	$.25\% \pm 1.28$
Off-policy REINFORCE Learned Reward	2.84 ± 0.23	34.56 ± 11.55	$2.79\% \pm 0.76$	$6.90\% \pm 1.45$
*Q-learning AMT**	$\mathbf{3.15 \pm 0.20}$	30.26 ± 4.64	$3.75\% \pm 0.93$	$\mathbf{5.41\% \pm 1.16}$

we take two steps to reduce confounding factors and correlation between users. First, during each A/B testing experiment, we evaluate all policies of interest simultaneously. This ensures that we have approximately the same number of users interacting with each policy w.r.t. time of day and weekday. This minimizes the effect of changes in the user distribution on the final user scores *within* that period. However, since the user distribution changes between the A/B testing experiments, we still cannot compare policy performance *across* A/B testing experiments. Second, we discard scores from returning users (i.e. users who have already evaluated the system once). Users who are returning to the system are likely to be influenced by their previous interactions with the system. For example, users who previously had a positive experience with the system may be biased towards giving high scores in their next interaction. Further, the users who return to the system are more likely to belong to a different user population than the users who only try the system once. This group of users may inherently have more free time and be more willing to engage with socialbots than other users. Discarding returning user scores ensures that the evaluation is not biased towards a subpopulation of users. By discarding scores from returning users, we also ensure that the evaluation counts every user exactly once. Finally, it should be noted that we ignore dialogues where the Alexa user did not give a score. This inevitably biases our evaluation, since users who do not provide a score are likely to have been dissatisfied with the system or to have been expecting different functionality (e.g. *non-conversational activities*, such as playing music, playing games or taking quizzes). One potential remedy is to have all dialogues evaluated by a third-party (e.g. by asking human annotators on Amazon Mechanical Turk to evaluate the dialogue), but that is beyond the scope of this work.

5.1 A/B Testing Experiment #1

The first A/B testing experiment was carried out between July 29th, 2017 and August 6th, 2017. We tested six dialogue manager policies: *Evibot + Alicebot*, *Supervised AMT*, *Supervised Learned Reward*, *Off-policy REINFORCE*, *Off-policy REINFORCE Learned Reward* and *Q-learning AMT*. For *Off-policy REINFORCE* and *Off-policy REINFORCE Learned Reward*, we use the greedy variant defined in eq. (5).

This experiment occurred early in the Amazon Alexa Prize competition. This means that Alexa users have few expectations towards our system (e.g. expectations that the system can converse on a particular topic, or that the system can engage in *non-conversational activities*, such as playing word games or taking quizzes). Further, the period July 29th - August 6th overlaps with the summer holidays in the United States. This means that we might expect more children to interact with system than during other seasons.

Policy Evaluation The results are given in Table 5.[26] The table shows the average Alexa user scores, average dialogue length, average percentage of positive user utterances and average percentage of negative user utterances. In total, over a thousand user ratings were collected after discarding returning users.[27] Each policy was evaluated by about two hundred unique Alexa users.

[26] 95% confidence intervals are computed under the assumption that the Alexa user scores for each policy are drawn from a normal distribution with its own mean and variance. This is an approximation, since the Alexa user scores only have support on the interval $[1, 5]$.

[27] Ratings were collected after the end of the semi-finals competition, where all ratings had been transcribed by human annotators.

As expected from our preliminary evaluation, we observe that *Q-learning AMT* and *Off-policy REINFORCE* perform best among all policies w.r.t. user scores. *Q-learning AMT* obtained an average user score of 3.15, which is significantly higher than all other policies at a 95% statistical significance level w.r.t. a one-tailed two-sample t-test. Interestingly, *Off-policy REINFORCE* achieved the longest dialogues with an average of $37.02/2 = 18.51$ turns per dialogue. This suggests *Off-policy REINFORCE* yields highly interactive and engaging conversations. However, *Off-policy REINFORCE* also had a slightly higher percentage of user utterances with negative sentiment compared to *Q-learning AMT*. This potentially indicates that the longer dialogues also include some frustrated interactions (e.g. users who repeat the same questions or statements in the hope that the system will return a more interesting response next time). The remaining policies achieved average Alexa user scores between 2.74 and 2.86, with the heuristic policy *Evibot + Alicebot* obtaining 2.86. This suggests that the other policies have not learned to select responses more appropriately than the *Evibot + Alicebot* heuristic.

In conclusion, the results indicate that the *risk tolerant* learned by the *Q-learning AMT* policy performs best among all policies. This shows that learning a policy through simulations in an *Abstract Discourse MDP* may serve as a fruitful path towards developing open-domain socialbots. In addition, the performance of *Off-policy REINFORCE* indicates that optimizing the policy directly towards Alexa user scores could also potentially yield improvements. However, further investigation is required.

Initiatorbot Evaluation This experiment also allowed us to analyze the outcomes of different conversation starter phrases given by the *Initiatorbot*. We carried out this analysis by computing the average Alexa user score for each of the 40 possible phrases. We found that phrases related to news (e.g. *"Do you follow the news?"*), politics (e.g. *"Do you want to talk about politics?"*) and travelling (e.g. *"Tell me, where do you like to go on vacation?"*) performed poorly across all policies. On the other hand, phrases related to animals (e.g. *"Do you have pets?"* and *"What is the cutest animal you can think of?"*), movies (e.g. *"Let's talk about movies. What's the last movie you watched?"*) and food (e.g. *"Let's talk about food. What is your favorite food?"*) performed well across all policies. For example, conversations where the *Initiatorbot* asked questions related to news and politics had an average Alexa user score of only 2.91 for the top two systems (*Off-policy REINFORCE* and *Q-learning AMT*). Mean while, conversations where the *Initiatorbot* asked questions about animals, food and movies the corresponding average Alexa user score was 3.17. We expected the conversation topic to affect user engagement, however it is surprising that these particular topics (animals, food and movies) were the most preferred ones. One possible explanation is that our system does not perform well on news, politics and travelling topics. However, the system already had several response models dedicated to discussing news and politics: six sequence-to-sequence models extracting responses from Reddit news and Reddit politics, two models extracting responses from Washington Post user comments and the *BoWTrump* model extracting responses from Donald J. Trump's Twitter profile. In addition, *Evibot* is capable of answering many factual questions about news and politics and *BoWFactGenerator* contains hundreds of facts related to news and politics. As such, there may be another more plausible explanation for users' preferences towards topics, such as animals, movies and food. One likely explanation is the age group of the users. While inspecting our logs, we observed that many children were interacting with our system, and it would hardly come as a surprise that these children may prefer to talk about animals, movies and foods rather than news, politics and travels.

5.2 A/B Testing Experiment #2

The second A/B testing experiment was carried out between August 6th, 2017 and August 13th, 2017. We tested two dialogue manager policies: *Off-policy REINFORCE* and *Q-learning AMT*. As before, we use the greedy variant of *Off-policy REINFORCE* defined in eq. (5).

This experiment occurred at the end of the Amazon Alexa Prize competition. This means that many Alexa users have already interacted with other socialbots in the competition, and therefore are likely to have developed expectations towards the systems. These expectations are likely to involve conversing on a particular topic or engaging in *non-conversational activities*, such as playing games). Further, the period August 6th - August 13th overlaps with the end of the summer holidays and the beginning of the school year in the United States. This means that we should expect less children to interact with the system than in the previous A/B testing experiment.

Table 6: Second A/B testing experiment with two different policies (± 95% confidence intervals). Star * indicates policy is significantly better than other policies at 95% statistical significance level.

Policy	User score	Dialogue length	Pos. utterances	Neg. utterances
*Off-policy REINFORCE**	$\mathbf{3.10 \pm 0.12}$	$\mathbf{34.86 \pm 3.66}$	$3.21\% \pm 0.44$	$7.96\% \pm 0.82$
Q-learning AMT	2.93 ± 0.12	32.06 ± 3.58	$\mathbf{3.42\% \pm 0.48}$	$\mathbf{7.56\% \pm 0.81}$

Policy Evaluation The results are given in Table 6. In total, about eight hundred user ratings were collected after discarding returning users. As such, each policy was evaluated by about four hundred unique Alexa users.

We observe that both *Off-policy REINFORCE* and *Q-learning AMT* perform better than the policies in the previous experiment. However, in this experiment, *Off-policy REINFORCE* achieved an average Alexa user score of 3.06 while *Q-learning AMT* achieved a lower score of only 2.95. Nonetheless, *Off-policy REINFORCE* is not statistically significantly better. Further, *Off-policy REINFORCE* achieves a slightly higher percentage of negative user utterances and a slightly lower percentage of positive user utterances compared *Q-learning AMT*.

As discussed earlier, the performance difference compared to the previous A/B testing experiment could be due to the change in user profiles and user expectations. At this point in time, more of the Alexa users have interacted with socialbots from other teams. Mean while, all socialbots have been evolving. Therefore, user expectations towards our system are likely to be higher now. Further, since the summer holidays have ended, less children and more adults are expected to interact with our system. It is plausible that these adults also have higher expectations towards the system, and even more likely that they are less playful and less tolerant towards mistakes. Given this change in user profiles and expectations, the *risk tolerant strategy* learned by the *Q-learning AMT* policy is likely to fare poorly compared to the *risk averse* strategy learned by *Off-policy REINFORCE*.

5.3 Discussion

The two policies *Q-learning AMT* and *Off-policy REINFORCE* have demonstrated substantial improvements over all other policies, including policies learned using supervised learning and heuristic policies. As discussed earlier, the *Q-learning AMT* policy achieved an average Alexa user score comparable to some of the top teams in the competition semi-finals. Mean while, *Off-policy REINFORCE* a very high number of turns in the dialogue, suggesting that the resulting conversations are far more interactive and engaging. The results demonstrate the advantages of the overall ensemble approach, where many different models generate natural language responses and the system policy selects one response among them. The results also highlight the advantages of learning the policy using reinforcement learning techniques. By optimizing the policy to maximize with either real-world user scores or rewards in the *Abstract Discourse MDP*, with a proxy reward function, we have demonstrated that significant gains can be achieved w.r.t. both real-world user scores and number of dialogue turns.

6 Future Work

6.1 Personalization

One important direction for future research is personalization, i.e. building a model of each user's personality, opinions and interests. This will allow the system to provide a better user experience by adapting the response models to known attributes of the user. We are in the process of implementing a state machine that given a user id, retrieves the relevant information attributes of the user from a database. If a particular user attribute is missing, then the state machine will ask the user for the relevant information and store it in the database. One important user attribute is the user's name. If no name is found in the database, the state machine may ask the user what they would like to be called and afterwards extracts the name from the user's response. If a personal name is detected, it is stored in the database to be available for other modules to insert into their responses. Name detection proceeds as follows. First we match the response against a small collection of templates, such as "my name is ..." or "call me ...". Then we use part-of-speech (POS) tags of the resulting matches to detect the end boundary of the name. To avoid clipping the name too early due to wrong POS tags, we also match words against a list of common names in the 1990 US Census data[28].

In the future, we plan to explore learning user embeddings from previous interactions with each user. This will allow the system to become more personalized, by providing our response models with additional context beyond the immediate dialogue history.

7 Conclusion

We have proposed a new large-scale ensemble-based dialogue system framework for the Amazon Alexa Prize competition. Our system leverages a variety of machine learning techniques, including deep learning and reinforcement learning. We have developed a new set of deep learning models for natural language retrieval and generation, including recurrent neural networks, sequence-to-sequence models and latent variable models. Further, we have developed a novel reinforcement learning procedure and evaluated it against existing reinforcement learning methods in A/B testing experiments with real-world users. These innovations have enabled us to make substantial improvements upon our baseline system. On a scale $1-5$, our best performing system reached an average user score of 3.15, with a minimal amount of hand-crafted states and rules and without engaging in *non-conversational activities* (such as playing games). This is comparable to some of the top systems in the semi-finals. Furthermore, the same system averaged a high $14.5-16.0$ turns per conversation, which suggests that our system is one of the most *interactive* and *engaging* systems in the competition. Since nearly all our system components are trainable machine learning models, the system is likely to improve greatly with more interactions and additional data.

Acknowledgments

We thank Aaron Courville, Michael Noseworthy, Nicolas Angelard-Gontier, Ryan Lowe, Prasanna Parthasarathi and Peter Henderson for helpful advice related to the system architecture, crowdsourcing and reinforcement learning throughout the Alexa Prize competition. We thank Christian Droulers for building the graphical user interface for text-based chat. We thank Amazon for providing Tesla K80 GPUs through the Amazon Web Services platform. Some of the Titan X GPUs used for this research were donated by the NVIDIA Corporation. The authors acknowledge NSERC, Canada Research Chairs, CIFAR, IBM Research, Nuance Foundation, Microsoft Maluuba and Druide Informatique Inc. for funding.

References

Ameixa, D., Coheur, L., Fialho, P. & Quaresma, P. (2014), Luke, I am your father: dealing with out-of-domain requests by using movies subtitles, *in* 'Intelligent Virtual Agents', Springer.

Aust, H., Oerder, M., Seide, F. & Steinbiss, V. (1995), 'The philips automatic train timetable information system', *Speech Communication* **17**(3).

Bird, S., Klein, E. & Loper, E. (2009), *Natural Language Processing with Python*, O'Reilly Media.

Blunsom, P., Grefenstette, E. & Kalchbrenner, N. (2014), A convolutional neural network for modelling sentences, *in* 'Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics', Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics.

Bohus, D., Raux, A., Harris, T. K., Eskenazi, M. & Rudnicky, A. I. (2007), Olympus: an open-source framework for conversational spoken language interface research, *in* 'Proceedings of the workshop on bridging the gap: Academic and industrial research in dialog technologies', Association for Computational Linguistics, pp. 32–39.

Breiman, L. (1996), 'Bagging predictors', *Machine learning* **24**(2), 123–140.

[28] Obtained from: `https://deron.meranda.us/data/`.

Charras, F., Duplessis, G. D., Letard, V., Ligozat, A.-L. & Rosset, S. (2016), Comparing system-response retrieval models for open-domain and casual conversational agent, *in* 'Workshop on Chatbots and Conversational Agent Technologies'.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H. & Bengio, Y. (2014), Learning phrase representations using rnn encoder–decoder for statistical machine translation, *in* 'EMNLP'.

Colby, K. M. (1981), 'Modeling a paranoid mind', *Behavioral and Brain Sciences* **4**.

Ferrucci, D., Brown, E., Chu-Carroll, J., Fan, J., Gondek, D., Kalyanpur, A. A., Lally, A., Murdock, J. W., Nyberg, E., Prager, J. et al. (2010), 'Building watson: An overview of the deepqa project', *AI magazine* **31**(3).

Glorot, X., Bordes, A. & Bengio, Y. (2011), Deep sparse rectifier neural networks, *in* 'Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics', pp. 315–323.

Im, J. (2017).
URL: *http://search.aifounded.com/*

Jurčíček, F., Dušek, O., Plátek, O. & Žilka, L. (2014), Alex: A statistical dialogue systems framework, *in* 'International Conference on Text, Speech, and Dialogue', Springer, pp. 587–594.

Kingma, D. & Ba, J. (2015), Adam: A method for stochastic optimization, *in* 'ICLR'.

Kingma, D. P. & Welling, M. (2014), 'Auto-encoding variational Bayes', *ICLR* .

Kiros, R., Zhu, Y., Salakhutdinov, R. R., Zemel, R., Urtasun, R., Torralba, A. & Fidler, S. (2015), Skip-thought vectors, *in* 'NIPS'.

Koren, Y., Bell, R. & Volinsky, C. (2009), 'Matrix factorization techniques for recommender systems', *Computer* **42**(8).

Lin, L.-J. (1993), Reinforcement learning for robots using neural networks, Technical report, Carnegie-Mellon Univ Pittsburgh PA School of Computer Science.

Liu, C.-W., Lowe, R., Serban, I. V., Noseworthy, M., Charlin, L. & Pineau, J. (2016), 'How NOT to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation', *arXiv:1603.08023* .

Lowe, R., Noseworthy, M., Serban, I. V., Angelard-Gontier, N., Bengio, Y. & Pineau, J. (2017), Towards an automatic Turing test: Learning to evaluate dialogue responses, *in* 'ACL'.

Lowe, R., Pow, N., Serban, I. & Pineau, J. (2015), The Ubuntu Dialogue Corpus: A Large Dataset for Research in Unstructured Multi-Turn Dialogue Systems, *in* 'SIGDIAL'.

Marelli, M., Bentivogli, L., Baroni, M., Bernardi, R., Menini, S. & Zamparelli, R. (2014), Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment., *in* 'SemEval Workshop, COLING'.

McGlashan, S., Fraser, N., Gilbert, N., Bilange, E., Heisterkamp, P. & Youd, N. (1992), Dialogue management for telephone information systems, *in* 'ANLC'.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S. & Dean, J. (2013), Distributed representations of words and phrases and their compositionality, *in* 'NIPS'.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D. & Riedmiller, M. (2013), 'Playing atari with deep reinforcement learning', *arXiv preprint arXiv:1312.5602* .

Nair, V. & Hinton, G. E. (2010), Rectified linear units improve restricted boltzmann machines, *in* 'Proceedings of the 27th international conference on machine learning (ICML-10)', pp. 807–814.

Ng, A. Y., Harada, D. & Russell, S. (1999), Policy invariance under reward transformations: Theory and application to reward shaping, *in* 'ICML', Vol. 99, pp. 278–287.

Nguyen, T., Rosenberg, M., Song, X., Gao, J., Tiwary, S., Majumder, R. & Deng, L. (2016), 'MS MARCO: A Human Generated MAchine Reading COmprehension Dataset', *arXiv preprint arXiv:1611.09268* .

Pennington, J., Socher, R. & Manning, C. D. (2014), Glove: Global vectors for word representation., *in* 'EMNLP', Vol. 14.

Precup, D. (2000), 'Eligibility traces for off-policy policy evaluation', *Computer Science Department Faculty Publication Series* .

Precup, D., Sutton, R. S. & Dasgupta, S. (2001), Off-policy temporal-difference learning with function approximation, *in* 'ICML'.

Rezende, D. J., Mohamed, S. & Wierstra, D. (2014), Stochastic backpropagation and approximate inference in deep generative models, *in* 'ICML', pp. 1278–1286.

Serban, I. V., Sordoni, A., Lowe, R., Charlin, L., Pineau, J., Courville, A. & Bengio, Y. (2017), A Hierarchical Latent Variable Encoder-Decoder Model for Generating Dialogues, *in* 'AAAI'.

Shawar, B. A. & Atwell, E. (2007), Chatbots: are they really useful?, *in* 'LDV Forum', Vol. 22.

Simpson, A. & Eraser, N. M. (1993), Black box and glass box evaluation of the sundial system, *in* 'Third European Conference on Speech Communication and Technology'.

Socher, R., Perelygin, A., Wu, J. Y., Chuang, J., Manning, C. D., Ng, A. Y., Potts, C. et al. (2013), Recursive deep models for semantic compositionality over a sentiment treebank, *in* 'Proceedings of the conference on empirical methods in natural language processing (EMNLP)', Vol. 1631, p. 1642.

Stolcke, A., Ries, K., Coccaro, N., Shriberg, E., Bates, R., Jurafsky, D., Taylor, P., Martin, R., Van Ess-Dykema, C. & Meteer, M. (2000), 'Dialogue act modeling for automatic tagging and recognition of conversational speech', *Computational linguistics* **26**(3).

Stone, B. & Soper, S. (2014), 'Amazon Unveils a Listening, Talking, Music-Playing Speaker for Your Home', *Bloomberg L.P* . Retrieved 2014-11-07.

Suendermann-Oeft, D., Ramanarayanan, V., Teckenbrock, M., Neutatz, F. & Schmidt, D. (2015), Halef: An open-source standard-compliant telephony-based modular spoken dialog system: A review and an outlook, *in* 'Natural language dialog systems and intelligent assistants', Springer.

Sutton, R. S. & Barto, A. G. (1998), *Reinforcement learning: An introduction*, number 1 *in* '1', MIT Press Cambridge.

Wallace, R. S. (2009), 'The anatomy of alice', *Parsing the Turing Test* .

Weizenbaum, J. (1966), 'Eliza—a computer program for the study of natural language communication between man and machine', *ACM* **9**(1).

Williams, J. D. (2011), An empirical evaluation of a statistical dialog system in public use, *in* 'Proceedings of the SIGDIAL 2011 Conference', Association for Computational Linguistics, pp. 130–141.

Williams, J. D., Raux, A. & Henderson, M. (2016), 'Introduction to the special issue on dialogue state tracking', *Dialogue & Discourse* **7**(3), 1–3.

Williams, J., Raux, A., Ramachandran, D. & Black, A. (2013), The dialog state tracking challenge, *in* 'SIGDIAL', pp. 404–413.

Williams, R. J. (1992), 'Simple statistical gradient-following algorithms for connectionist reinforcement learning', *Machine learning* **8**(3-4).

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K. et al. (2016), 'Google's neural machine translation system: Bridging the gap between human and machine translation', *arXiv preprint arXiv:1609.08144* .

Yu, L., Hermann, K. M., Blunsom, P. & Pulman, S. (2014), Deep learning for answer sentence selection, *in* 'NIPS, Workshop on Deep Learning'.

Yu, Z., Xu, Z., Black, A. W. & Rudnicky, A. I. (2016), Strategy and policy learning for non-task-oriented conversational systems., *in* 'SIGDIAL'.

Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A. & Fidler, S. (2015), Aligning books and movies: Towards story-like visual explanations by watching movies and reading books, *in* 'ICCV'.