The Octopus Approach to the Alexa Competition: A Deep Ensemble-based Socialbot

Iulian V. Serban, **Chinnadhurai Sankar**, **Saizheng Zhang**, **Zhouhan Lin**, **Sandeep Subramanian**, **Taesup Kim**, **Sarath Chandar**, **Nan Rosemary Ke**, **Sai Rajeswar**, **Alexandre de Brebisson**, **Jose M. R. Sotelo**, **Dendi Suhubdy**, **Vincent Michalski**, **Alexandre Nguyen** and **Yoshua Bengio**[1]
University of Montreal, Montreal, Quebec, Canada

Abstract

We present a novel, large-scale ensemble-based system for the Amazon Alexa Prize competition. Our system leverages state-of-the-art methods from deep learning and reinforcement learning. We carry out A/B testing experiments with real-world users and demonstrate that our approach yields substantial improvements over several baseline systems. During the competition semi-finals, our best performing system obtained a substantially higher average Alexa user score and number of back-and-forth turns compared to the average of all teams. Due to its machine learning architecture, our system is likely to improve with additional data.

1 Introduction

Conversational agents and dialogue systems - including chatbots and personal assistants - are becoming ubiquitous in modern society. In 2016, Amazon 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. This article describes the system developed by the *MILA Team* at University of Montreal.

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. Further, we apply reinforcement learning — including methods based on value functions and policy gradients — to intelligently combine an ensemble of retrieval and generation models. In particular, we propose a novel reinforcement learning procedure, which yields substantial improvements in A/B testing experiments with real-world users.

On a scale $1-5$, our best performing system reached an average user score of 3.15 in the semi-finals (experiment conducted 07/29/2017 – 08/06/2017).[2] Furthermore, our best performing system averaged $14.5-16.0$ turns per conversation. In comparison, all teams in the competition semi-finals achieved an average user score of only 2.92 and an average of 11 turns per conversation (07/01/2017 – 08/30/2017). These results indicate that our system is highly *competitive* and *engaging*. In contrast to some other teams, our system achieved this without engaging in *non-conversational activities* (e.g. playing games or quizzes) and by relying on a minimal number of hand-crafted rules and states. As nearly all system components are learnable, the system is likely to improve greatly with additional data.

[1] CIFAR Fellow.

[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 these experiments, not on the leaderboard which averages the scores of all our systems.

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

2 System Overview

Early work on dialogue systems [Weizenbaum, 1966, Aust et al., 1995, McGlashan et al., 1992] were based mainly on states and rules hand-crafted by human experts. Modern dialogue systems typically follow a hybrid architecture, which combines hand-crafted states and rules with statistical machine learning algorithms [Suendermann-Oeft et al., 2015]. Due to the complexity of human language, however, it is impossible to enumerate all of the 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 language. As such, the system utilizes only a small number of hand-crafted states and rules. Meanwhile, every system component has been designed to be optimized (trained) using machine learning algorithms. By optimizing these system components first independently on massive datasets and then jointly on real-world user interactions, the system will learn implicitly all relevant states and rules for conducting open-domain conversations. Given an adequate amount of examples, such a system should outperform any system based on states and rules hand-crafted by human experts. Further, the system will continue to improve in perpetuity with additional data.

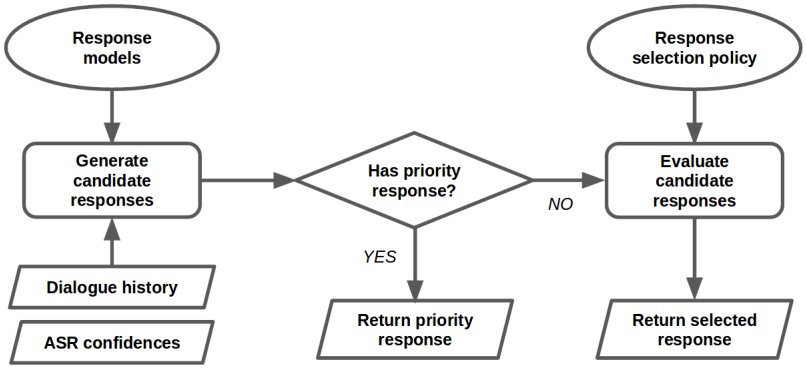

Figure 1: Dialogue manager control flow.

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], the IBM Watson question-answering system [Ferrucci et al., 2010] and Google's machine translation system [Wu et al., 2016].

Our system consists of an ensemble of response models (see Figure 1). Each response model takes as input a dialogue history and outputs a response in natural language text. 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. The *dialogue manager* is responsible for combining the response models together. As input, the dialogue manager 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 dialogue manager follows 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. 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.

3 Response Models

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

Table 1: Example dialogues and candidate responses generated by response models. The chosen system response is marked in bold.

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

Template-based Models: The *Alicebot* model uses a set of AIML (artificial intelligence markup language) templates to produce a response given the dialogue history [Wallace, 2009]. We use the Alice kernel available at `www.alicebot.org`. We modify specific templates to output priority responses when the socialbot is asked about its name, age and location.

The *Elizabot* model performs string matching to select an answer from a set of templates [Weizenbaum, 1966], similar to *Alicebot*.[3] It was designed to mimic a Rogerian psychotherapist. Most of *Elizabot*'s responses are personal questions, which are meant to engage the user to continue the conversation.

The *Initiatorbot* model acts as a *conversation starter*: it asks the user an open-ended question to get the conversation started and increase the user's engagement. 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?"*. Further, the model can also start the conversation by stating an *interesting* or *fun* fact. The facts are taken from the *BoWFactGenerator* model, described later.

The *Storybot* model outputs a short fiction story at the request of the user. This model was implemented after observing that many users were asking the socialbot to tell stories.[4] *Storybot* determines if the user requested a story by checking for keywords, such as *tell* and *story*. 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. The rest of the system will not encourage the user to request stories.

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

[4] Requests to tell stories might be a side-effect of users interacting with other Alexa applications on socialbots from other teams, some of which emphasize *non-conversational activities*, such as story-telling and game-playing.

Knowledge Base Question Answering Models: The *Evibot* response model forwards the user utterances to Evi, Amazon's question-answering web service: `www.evi.com`. Evi was designed primarily to handle factual questions. Therefore, if the user utterance contains a wh-word (e.g. *"who"*, *"what"*) the model returns a priority response. Otherwise, it returns a non-priority response.

The *BoWMovies* model is knowledge base response model, which responds to questions related to movies, actors and directors. It identifies known entity names (e.g. movie titles and actor names) and known entity tags within a user utterance using similarity w.r.t. Word2Vec word embeddings [Mikolov et al., 2013] (i.e. similarity to keywords related to information slots in the databases, such as *movie plot* and *release year*). If an entity name and a tag are present, the agent dispatches an API call to retrieve the relevant data item from one of three databases: *OMDB*, *GoogleKnowledgeGraph* and *Wikidata*.

Retrieval-based Neural Networks: The system contains several VHRED models trained on different datasets [Serban et al., 2017b]. 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 word embeddings [Pennington et al., 2014].[5] An approximate log-likelihood score is computed for each of the K responses, and the response with the highest likelihood is returned. The system has 4 models based on datasets scraped from Reddit, one model based on news articles and one model based on movie subtitles:

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

In addition, 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 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. This model is called *SkipThoughtBooks*.

Furthermore, the system contains two retrieval models based on the Dual Encoder model [Lowe et al., 2015, 2017b], *DualEncoderRedditPolitics* and *DualEncoderRedditNews*. Similar to the VHRED models, first a set of $K = 50$ candidate responses are retrieved using TF-IDF cosine similarity based on word embeddings. Then, both models score the candidate responses and return the response with the highest score. The model *DualEncoderRedditPolitics* was trained on and retrieves candidate responses from Reddit `https://www.reddit.com/r/politics`. Similarly, the model *DualEncoderRedditNews* was trained on and retrieves candidate responses from Reddit `https://www.reddit.com/r/news`.

Finally, the system contains three retrieval models based on bag-of-words TF-IDF pre-trained word embeddings [Pennington et al., 2014, Mikolov et al., 2013]. These models retrieve the response with the highest cosine similarity. The *BoWWashingtonPost* model retrieves user comments from WashingtonPost news articles using 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 embeddings. It only returns a response when at least one relevant keyword is found in the user's utterance (e.g. when the word *"Trump"* is mentioned by the user). The next model *BoWFactGenerator* retrieves responses from a set of about 2500 *fun* facts, including facts about animals, geography and history, using pre-trained Word2Vec embeddings.

Retrieval-based Logistic Regression: 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 model to select its response.

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

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 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 through an MLP to predict if the snippet is a good response to the utterance. The model is trained on the Microsoft Marco dataset to predict the relevancy of a snippet given a user query [Nguyen et al., 2016] .

Generation-based Neural Networks: The system contains a generative recurrent neural network language model, called *GRUQuestionGenerator*, 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 is trained on Reddit Politics and Reddit News conversations, wherein posts were labeled as questions by detecting question marks. For generation, we first condition the model on a short question template (e.g. *"How about"*), and then generate the rest of the question by sampling from the model. Several candidate responses are sampled, and the response with the highest log-likelihood is returned.

4 Model Selection Policy

After generating the candidate response set, the dialogue manager uses a *model selection policy* to select the response it returns to the user. The dialogue manager 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 dialogue manager 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 dialogue manager chooses to respond with a short news statement, the user may be less pleased for one turn. However, this 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, which are evaluated with real-world users in the next section.

We use reinforcement learning [Sutton and Barto, 1998]. The dialogue manager is an agent, which takes actions in an environment in order to maximize rewards. For each time step $t=1,...,T$, the agent observes the dialogue history h_t and must choose one of K actions (responses): $a_t^1,...,a_t^K$. After taking an action, it receives a reward r_t and is transferred to the next state h_{t+1} (which includes the action and the user's next response) where it is provided with a new set of K actions: $a_{t+1}^1,...,a_{t+1}^K$. The agent must maximize the discounted sum of rewards, $R=\sum_{t=1}^{T}\gamma^t r_t$, where $\gamma \in (0,1]$ is a discount factor.

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:

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

which estimates the expected discounted sum of rewards – referred to as the *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:

$$\pi_\theta(h_t) = \underset{k}{\operatorname{argmax}}\, Q_\theta(h_t,a_t^k). \tag{2}$$

This approach is related to recent work by Lowe et al. [2017a] and Yu et al. [2016].

Stochastic Policy Parametrization: This approach instead parameterizes a distribution over actions:

$$\pi_\theta(a_t^k|h_t) = \frac{e^{\lambda^{-1} f_\theta(h_t,a_t^k)}}{\sum_{a_t'} e^{\lambda^{-1} f_\theta(h_t,a_t')}} \quad \text{for } k=1,...,K, \tag{3}$$

where $f_\theta(h_t,a_t^k)$ is the *scoring function*, parametrized by θ, which assigns a scalar score to each response a_t^k conditioned on h_t. The parameter λ controls the entropy of the distribution. The stochastic policy can be transformed to a deterministic (greedy) policy by selecting the action with highest probability:

$$\pi_\theta^{\text{greedy}}(h_t) = \underset{k}{\operatorname{argmax}}\, \pi_\theta(a_t^k|h_t). \tag{4}$$

We parametrize the scoring function and action-value function as neural networks with five layers. The first layer is the input, consisting of 1458 features representing both the dialogue history, h_t, and the candidate response, a_t. These features are based on a combination of word embeddings, dialogue acts [Stolcke et al., 2000], part-of-speech tags [Bird et al., 2009], unigram word overlap, bigram word overlap and model-specific features.[6] The second layer contains 500 hidden units, computed by applying a linear transformation followed by the rectified linear activation function to the input layer features. The third layer contains 20 hidden units, computed by applying a linear transformation to the preceding layer units. The fourth layer contains 5 outputs units, which are probabilities (i.e. all values are positive and their sum equals 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, described later. The fifth layer is the final output layer, which is a single scalar value computed by applying a linear transformation to the units in the third and fourth layers. In order to learn the parameters, we use five different machine learning approaches described next.

Supervised Learning with Crowdsourced Labels: The first approach to learning the policy parameters is called *Supervised Learning AMT*. This approach estimates the action-value function Q_θ using supervised learning on crowdsourced labels. This approach also serves as initialization for the other approaches discussed later. It also serves as initialization for all other approaches.

We use Amazon Mechanical Turk (AMT) to collect data for training the policy. 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 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. As examples, we use a few thousand dialogues recorded between Alexa users and a preliminary version of the systems. In total, we collected 199,678 labels, which are split this into training (train), development (dev) and testing (test) datasets consisting of respectively 137,549, 23,298 and 38,831 labels each.

We optimize the model parameters θ w.r.t. log-likelihood (cross-entropy) using mini-batch stochastic gradient descent (SGD) to predict the 4th layer, which represents the AMT labels. Since we do not have labels for the last layer of the model, we fix the corresponding linear transformation to $[1.0,2.0,3.0,4.0,5.0]$. In other words, we assign a score of 1.0 for an inappropriate response, 3.0 for an acceptable response and 5.0 for an excellent response.

Off-policy REINFORCE: Our next approach learns a stochastic policy directly from examples of dialogues recorded between the system and real-world users. Let $\{h_t^d, a_t^d, R^d\}_{d,t}$ be a set of examples, where d indicates the dialogue and t indicates the time step (turn). Let h_t^d be the dialogue history, a_t^d be the agent's action and R^d be the observed return. Further, let θ_d be the parameters of the stochastic policy π_{θ_t} used during dialogue d. We use a re-weighted variant of the *REINFORCE* algorithm [Williams, 1992, Precup, 2000, Precup et al., 2001], with learning rate $\alpha > 0$, which updates the policy parameters for each example (h_t^d, a_t^d, R^d):

$$\Delta\theta = \alpha \frac{\pi_\theta(a_t^d|h_t^d)}{\pi_{\theta_d}(a_t^d|h_t^d)} \nabla_\theta \log \pi_\theta(a_t^d|h_t^d)\, R^d. \tag{5}$$

The intuition behind the algorithm is analogous to learning from trial and error. Examples with high user scores R^d will increase the probability of the actions taken by the agent through the term $\nabla_\theta \log \pi_\theta(a_t^d|h_t^d)\, R^d$. Vice versa, examples with low user scores will decrease the action probabilities. The ratio on the left-hand-side corrects for the discrepancy between the learned policy, π_θ, and the policy under which the data was collected, π_{θ_d}. We evaluate the policy using an estimate of the expected return:

$$\mathrm{E}_{\pi_\theta}[R] \approx \sum_{d,t} \frac{\pi_\theta(a_t^d|h_t^d)}{\pi_{\theta_d}(a_t^d|h_t^d)}\, R^d. \tag{6}$$

For training, we use over five thousand dialogues and scores collected in interactions between real users and a preliminary version of our system between June 30th to July 24th, 2017. We optimize the policy parameters on a training set with SGD based on eq. (5). We select hyper-parameters and early-stop on a development set based on eq. (6).

Learned Reward Function: Our two next approaches trains a linear regression model to predict the user score from a given dialogue. Given a dialogue history h_t and a candidate response a_t, the model

[6] To limit the effect of speech recognition errors in our experiments, ASR confidence features are not included.

g_ϕ, with parameters ϕ, predicts the corresponding user score. As training data is scarce, we only use 23 higher-level features as input. The model is trained on the same dataset as *Off-policy REINFORCE*.

The regression model g_ϕ is used in two ways. First, it is used to fine-tune the action-value function learned by *Supervised Learning AMT* to more accurately predict the user score. Specifically, the output layer is fine-tuned w.r.t. the squared-error between its own prediction and g_ϕ. This new policy is called *Supervised AMT Learned Reward*. Second, the regression model is combined with *Off-policy REINFORCE* into a policy called *Off-policy REINFORCE Learned Reward*. This policy is trained as *Off-policy REINFORCE*, but where R^d is replaced with the predicted user score g_ϕ in eq. (5).

Q-learning with the Abstract Discourse Markov Decision Process: Our final approach is based on learning a policy through a simplified Markov decision process (MDP), called the *Abstract Discourse MDP*. The MDP is fitted on the same dataset of dialogues and user scores as before. In particular, the reward function of the MDP is defined as the expected score predicted by *Supervised AMT*. For a description of the MDP, the reader is referred to the technical report by Serban et al. [2017a].

Given the *Abstract Discourse MDP*, we use *Q-learning* with *experience replay* to learn the policy with an action-value parametrization [Mnih et al., 2013, Lin, 1993]. We use an experience replay memory buffer of size 1000 and an ϵ-greedy exploration scheme with $\epsilon = 0.1$. We experiment with discount factors $\gamma \in \{0.1, 0.2, 0.5\}$. Training is based on SGD and carried out in two alternating phases. For every 100 episodes of training, we evaluate the policy over 100 episodes w.r.t. average return. During evaluation, the dialogue histories are sampled from a separate set of dialogue histories. This ensures that the policy is not *overfitting* the finite set of dialogue histories. We select the policy which performs best w.r.t. average return. This policy is called *Q-learning AMT*. A quantitative analysis shows that the learned policy is highly *risk tolerant* [Serban et al., 2017a].

5 A/B Testing Experiments

We carry out A/B testing experiments to evaluate the dialogue manager policies for selecting the response model. When an Alexa user starts a conversation with the system, they are assigned at random to a policy and afterwards the dialogue and their score is recorded.

A major issue with the A/B testing experiments is that the distribution of Alexa users changes through time. Different types of users will be using the system depending on the time of day, weekday and holiday season. In addition, user expectations towards our system change as users interact with other socialbots in the competition. Therefore, we must take steps to reduce confounding factors and correlations between users. First, during each A/B testing experiment, we simultaneously evaluate all policies of interest. 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 the changing user distribution *within* each A/B testing period. 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 influenced by their previous interactions with the system. For example, users who had a positive previous experience may be biased towards giving higher scores in their next interaction.

5.1 Experiment Periods

Exp #1: The first A/B testing experiment was conducted between July 29th and August 6th, 2017. We tested the dialogue manager policies *Supervised AMT*, *Supervised AMT Learned Reward*, *Off-policy REINFORCE*, *Off-policy REINFORCE Learned Reward* and *Q-learning AMT*. We used the greedy variants for the Off-policy REINFORCE policies. We also tested a heuristic baseline policy *Evibot + Alicebot*, which selects the *Evibot* model response if available, and otherwise selects the *Alicebot* model response. Over a thousand user scores were collected with about two hundred user scores per policy.[7]

This experiment occurred in the middle of the competition semi-finals. In this period, users are likely to have relatively few expectations towards the systems in the competition (e.g. that the system can converse on a particular topic or engage in *non-conversational activities*, such as playing games). Further, the period July 29th - August 6th overlaps with the summer holidays in the United States. As such, we might expect more children to interact with system here than during other seasons.

[7] To ensure high accuracy, human annotators were used to transcribe the audio related to the Alexa user scores for the first and second experiments. Amazon's speech recognition system was used in the third experiment.

Table 2: A/B testing results ($\pm$ 95% confidence intervals). Stars indicate statistical significance at 95%.

	Policy	User score	Dialogue length	Pos %	Neg %
Exp #1	*Evibot + Alicebot*	2.86±0.22	31.84±6.02	2.80%±0.79	5.63%±1.27
	Supervised AMT	2.80±0.21	34.94±8.07	**4.00%±1.05**	8.06%±1.38
	Supervised AMT Learned Reward	2.74±0.21	27.83±5.05	2.56%±0.70	6.46%±1.29
	Off-policy REINFORCE	2.86±0.21	**37.51±7.21**	3.98%±0.80	6.25±1.28
	Off-policy REINFORCE Learned Reward	2.84±0.23	34.56±11.55	2.79%±0.76	6.90%±1.45
	*Q-learning AMT**	**3.15±0.20**	30.26±4.64	3.75%±0.93	**5.41%±1.16**
Exp #2	*Off-policy REINFORCE*	**3.06±0.12**	**34.45±3.76**	3.23%±0.45	7.97%±0.85
	Q-learning AMT	2.92±0.12	31.84±3.69	**3.38%±0.50**	**7.61%±0.84**
Exp #3	*Off-policy REINFORCE*	3.03±0.18	30.93±4.96	2.72±0.59	7.36±1.22
	Q-learning AMT	**3.06±0.17**	**33.69±5.84**	**3.63±0.68**	**6.67±0.98**
Semi-finals average[8]	*All teams*	2.92	22	-	-
	Non-finalist teams	2.81	22	-	-
	Finalist teams	**3.31**	**26**	-	-

Exp #2: The second A/B testing experiment was conducted between August 6th and August 15th, 2017. We tested the two policies *Off-policy REINFORCE* and *Q-learning AMT*. Prior to beginning the experiment, minor system improvements were carried out w.r.t. the *Initiatorbot* and filtering out profanities. In total, about six hundred user scores were collected per policy.

This experiment occurred at the end of the competition semi-finals. At this point, many users have already interacted with other socialbots in the competition, and are therefore likely to have developed expectations towards the systems (e.g. conversing on a particular topic or engaging in *non-conversational activities*, such as playing games). Further, the period August 6th - August 15th overlaps with the end of the summer holidays and the beginning of the school year in the United States. This means we should expect less children interacting than in the previous A/B testing experiment.

Exp #3: The third A/B testing experiment was carried out between August 15th, 2017 and August 21st, 2017. Due to the surprising results in the previous A/B testing experiment, we decided to continue testing the two policies *Off-policy REINFORCE* and *Q-learning AMT*. In total, about three hundred user ratings were collected for each policy after discarding returning users.

This experiment occurred after the end of the competition semi-finals. This means that it is likely that many Alexa users have already developed expectations towards the systems. Further, the period August 15th - August 21st lies entirely within the beginning of the school year in the United States. We might expect less children to interact with the system than in the previous A/B testing experiment.

5.2 Results & Discussion

Table 2 shows the average Alexa user scores, average dialogue length, average percentage of positive user utterances and average percentage of negative user utterances.[9][10]

We observe that *Q-learning AMT* performed best among all policies w.r.t. Alexa user scores in the first and third experiments. In the first experiment, *Q-learning AMT* obtained an average user score of 3.15, which is significantly better than all other policies at a 95% significance level under a two-sample t-test. This is supported by the percentage of user utterances with positive and negative sentiment, where *Q-learning AMT* consistently obtained the lowest percentage of negative sentiment user utterances while maintaining a high percentage of positive sentiment user utterances. In comparison, the average user

[8] The aggregate scores for all, non-finalist and finalist teams are for the period 07/01/2017 – 08/30/2017.

[9] 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.

[10] The aggregate statistics for the teams in the semi-finals were reported by Amazon.

score for all the teams in the competition during the semi-finals was only 2.92. Next comes *Off-policy REINFORCE*, which performed best in the second experiment. In the second and third experiments, *Off-policy REINFORCE* also performed substantially better than all the other policies in the first experiment. Further, in the first experiment, *Off-policy REINFORCE* also obtained the longest dialogues with an average of $37.51/2 = 18.76$ turns per dialogue. In comparison, the average number of turns per dialogue for all the teams in the competition semi-finals (07/01/2017 – 08/30/2017) was only 11. This means *Off-policy REINFORCE* has over 70% more turns on average than the other teams in the competition semi-finals. The remaining policies achieved average user scores between 2.74 and 2.86, suggesting that they have not learned to select responses more appropriately than the heuristic policy *Evibot + Alicebot*.

In conclusion, the two policies *Q-learning AMT* and *Off-policy REINFORCE* have demonstrated substantial improvements over all other policies. Further, *Q-learning AMT* achieved an average Alexa user score and number of turns substantially above the average of all teams in the competition semi-finals. This strongly suggests that learning a policy through simulations in an *Abstract Discourse MDP* may serve as a fruitful path towards developing open-domain socialbots. Further, the performance of *Off-policy REINFORCE* suggests that optimizing the policy directly towards user scores also may serve as a fruitful path. In particular, *Off-policy REINFORCE* obtained a substantial increase in the average number of turns in the dialogue compared to the average of the teams in the semi-finals, suggesting that the resulting conversations are significantly more engaging. Overall, the experiments demonstrate the advantages of the ensemble approach, where many different models output natural language responses and the system policy selects one response among them.

6 Conclusion

We have proposed and evaluated a new large-scale ensemble-based dialogue system framework for the Amazon Alexa Prize competition. Our system leverages a variety of machine learning methods, 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 Amazon Alexa users. These innovations have enabled us to make substantial improvements upon our baseline system. Our best performing system reached an average user score of 3.15, on a scale $1-5$, with a minimal amount of hand-crafted states and rules and without engaging in *non-conversational activities* (such as playing games). This is substantially above the average score of all teams in the competition semi-finals, which was 2.92. Furthermore, the same system averaged $14.5-16.0$ turns per conversation, which was also substantially higher than the average number of turns of all teams in competition semi-finals. This improvement in back-and-forth exchanges between the user and system suggests that our system is one of the most *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 Mathieu Germain and Michael Pieper for helping with development and testing. We thank Joelle Pineau, Aaron Courville and other students for feedback. We thank Amazon for providing Tesla K80 GPUs. Some Titan X GPUs used for this research were donated by the NVIDIA Corporation.

References

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

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

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

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

R. Kiros, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler. Skip-thought vectors. In *NIPS*, 2015.

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

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

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

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

R. Lowe, M. Noseworthy, I. V. Serban, N. Angelard-Gontier, Y. Bengio, and J. Pineau. Towards an automatic Turing test: Learning to evaluate dialogue responses. In *ACL*, 2017a.

R. T. Lowe, N. Pow, I. V. Serban, L. Charlin, C.-W. Liu, and J. Pineau. Training end-to-end dialogue systems with the ubuntu dialogue corpus. *Dialogue & Discourse*, 8(1), 2017b.

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

S. McGlashan, N. Fraser, N. Gilbert, E. Bilange, P. Heisterkamp, and N. Youd. Dialogue management for telephone information systems. In *ANLC*, 1992.

T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, 2013.

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

T. Nguyen, M. Rosenberg, X. Song, J. Gao, S. Tiwary, R. Majumder, and L. Deng. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. *arXiv preprint arXiv:1611.09268*, 2016.

J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *EMNLP*, 2014.

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

D. Precup, R. S. Sutton, and S. Dasgupta. Off-policy temporal-difference learning with function approximation. In *ICML*, 2001.

I. V. Serban, C. Sankar, M. Germain, S. Zhang, Z. Lin, S. Subramanian, T. Kim, M. Pieper, S. Chandar, N. R. Ke, et al. A Deep Reinforcement Learning Chatbot. *arXiv preprint arXiv:1709.02349*, 2017a.

I. V. Serban, A. Sordoni, R. Lowe, L. Charlin, J. Pineau, A. Courville, and Y. Bengio. A Hierarchical Latent Variable Encoder-Decoder Model for Generating Dialogues. In *AAAI*, 2017b.

A. Stolcke, K. Ries, N. Coccaro, E. Shriberg, R. Bates, D. Jurafsky, et al. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational linguistics*, 26(3), 2000.

D. Suendermann-Oeft, V. Ramanarayanan, M. Teckenbrock, F. Neutatz, and D. Schmidt. 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, 2015.

R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT Press Cambridge, 1998.

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

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

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

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

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

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