Adversarial Learning for Neural Dialogue Generation

Jiwei Li[1], Will Monroe[1], Tianlin Shi[1], Sébastien Jean[2], Alan Ritter[3] and Dan Jurafsky[1]

[1]Stanford University, Stanford, CA, USA
[2]New York University, NY, USA
[3]Ohio State University, OH, USA
jiweil,wmonroe4,tianlins,jurafsky@stanford.edu
sebastien@cs.nyu.edu
ritter.1492@osu.edu

Abstract

In this paper, drawing intuition from the Turing test, we propose using adversarial training for open-domain dialogue generation: the system is trained to produce sequences that are indistinguishable from human-generated dialogue utterances. We cast the task as a reinforcement learning (RL) problem where we jointly train two systems, a generative model to produce response sequences, and a discriminator—analagous to the human evaluator in the Turing test— to distinguish between the human-generated dialogues and the machine-generated ones. The outputs from the discriminator are then used as rewards for the generative model, pushing the system to generate dialogues that mostly resemble human dialogues.

In addition to adversarial training we describe a model for adversarial *evaluation* that uses success in fooling an adversary as a dialogue evaluation metric, while avoiding a number of potential pitfalls. Experimental results on several metrics, including adversarial evaluation, demonstrate that the adversarially-trained system generates higher-quality responses than previous baselines.

1 Introduction

Open domain dialogue generation (Ritter et al., 2011; Sordoni et al., 2015; Xu et al., 2016; Wen et al., 2016; Li et al., 2016b; Serban et al., 2016c, 2017) aims at generating meaningful and coherent dialogue responses given the dialogue history. Prior systems, e.g., phrase-based machine translation systems (Ritter et al., 2011; Sordoni et al., 2015) or end-to-end neural systems (Shang et al., 2015; Vinyals and Le, 2015; Li et al., 2016a; Yao et al., 2015; Luan et al., 2016) approximate such a goal by predicting the next dialogue utterance given the dialogue history using the maximum likelihood estimation (MLE) objective. Despite its success, this over-simplified training objective leads to problems: responses are dull, generic (Sordoni et al., 2015; Serban et al., 2016a; Li et al., 2016a), repetitive, and short-sighted (Li et al., 2016d).

Solutions to these problems require answering a few fundamental questions: what are the crucial aspects that characterize an ideal conversation, how can we quantitatively measure them, and how can we incorporate them into a machine learning system? For example, Li et al. (2016d) manually define three ideal dialogue properties (ease of answering, informativeness and coherence) and use a reinforcement-learning framework to train the model to generate highly rewarded responses. Yu et al. (2016b) use keyword retrieval confidence as a reward. However, it is widely acknowledged that manually defined reward functions can't possibly cover all crucial aspects and can lead to suboptimal generated utterances.

A good dialogue model should generate utterances indistinguishable from human dialogues. Such a goal suggests a training objective resembling the idea of the Turing test (Turing, 1950). We borrow the idea of adversarial training (Goodfellow et al., 2014; Denton et al., 2015) in computer vision, in which we jointly train two models, a generator (a neural SEQ2SEQ model) that defines the probability of generating a dialogue sequence, and a discriminator that labels dialogues as human-generated or machine-generated. This discriminator is analogous to the evaluator in the Turing test. We cast the task as a reinforcement learning problem, in which the quality of machine-generated utterances is measured by its ability to fool the discriminator into believing that it is a

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2157–2169
Copenhagen, Denmark, September 7–11, 2017.

human-generated one. The output from the discriminator is used as a reward to the generator, pushing it to generate utterances indistinguishable from human-generated dialogues.

The idea of a Turing test—employing an evaluator to distinguish machine-generated texts from human-generated ones—can be applied not only to training but also testing, where it goes by the name of adversarial evaluation. Adversarial evaluation was first employed in Bowman et al. (2016) to evaluate sentence generation quality, and preliminarily studied for dialogue generation by Kannan and Vinyals (2016). In this paper, we discuss potential pitfalls of adversarial evaluations and necessary steps to avoid them and make evaluation reliable.

Experimental results demonstrate that our approach produces more interactive, interesting, and non-repetitive responses than standard SEQ2SEQ models trained using the MLE objective function.

2 Related Work

Dialogue generation Response generation for dialogue can be viewed as a source-to-target transduction problem. Ritter et al. (2011) frame the generation problem as a machine translation problem. Sordoni et al. (2015) improved Ritter et al.'s system by rescoring the outputs of a phrasal MT-based conversation system with a neural model incorporating prior context. Recent progress in SEQ2SEQ models have inspired several efforts (Vinyals and Le, 2015; Serban et al., 2016a,d; Luan et al., 2016) to build end-to-end conversational systems that first apply an encoder to map a message to a distributed vector representing its meaning and then generate a response from the vector.

Our work adapts the encoder-decoder model to RL training, and can thus be viewed as an extension of Li et al. (2016d), but with more general RL rewards. Li et al. (2016d) simulate dialogues between two virtual agents, using policy gradient methods to reward sequences that display three useful conversational properties: informativity, coherence, and ease of answering. Our work is also related to recent efforts to integrate the SEQ2SEQ and reinforcement learning paradigms, drawing on the advantages of both (Wen et al., 2016). For example, Su et al. (2016) combine reinforcement learning with neural generation on tasks with real users. Asghar et al. (2016) train an end-to-end RL dialogue model using human users.

Dialogue quality is traditionally evaluated (Sordoni et al., 2015, e.g.) using word-overlap metrics such as BLEU and METEOR scores used for machine translation. Some recent work (Liu et al., 2016) has started to look at more flexible and reliable evaluation metrics such as human-rating prediction (Lowe et al., 2017) and next utterance classification (Lowe et al., 2016).

Adversarial networks The idea of generative adversarial networks has enjoyed great success in computer vision (Radford et al., 2015; Chen et al., 2016a; Salimans et al., 2016). Training is formalized as a game in which the generative model is trained to generate outputs to fool the discriminator; the technique has been successfully applied to image generation.

However, to the best of our knowledge, this idea has not achieved comparable success in NLP. This is due to the fact that unlike in vision, text generation is discrete, which makes the error outputted from the discriminator hard to backpropagate to the generator. Some recent work has begun to address this issue: Lamb et al. (2016) propose providing the discriminator with the intermediate hidden vectors of the generator rather than its sequence outputs. Such a strategy makes the system differentiable and achieves promising results in tasks like character-level language modeling and handwriting generation. Yu et al. (2016a) use policy gradient reinforcement learning to backpropagate the error from the discriminator, showing improvement in multiple generation tasks such as poem generation, speech language generation and music generation. Outside of sequence generation, Chen et al. (2016b) apply the idea of adversarial training to sentiment analysis and Zhang et al. (2017) apply the idea to domain adaptation tasks.

Our work is distantly related to recent work that formalizes sequence generation as an action-taking problem in reinforcement learning. Ranzato et al. (2016) train RNN decoders in a SEQ2SEQ model using policy gradient to obtain competitive machine translation results. Bahdanau et al. (2017) take this a step further by training an actor-critic RL model for machine translation. Also related is recent work (Shen et al., 2016; Wiseman and Rush, 2016) to address the issues of exposure bias and loss-evaluation mismatch in neural translation.

3 Adversarial Training for Dialogue Generation

In this section, we describe in detail the components of the proposed adversarial reinforcement

learning model. The problem can be framed as follows: given a dialogue history x consisting of a sequence of dialogue utterances,[1] the model needs to generate a response $y = \{y_1, y_2, ..., y_T\}$. We view the process of sentence generation as a sequence of actions that are taken according to a policy defined by an encoder-decoder recurrent neural network.

3.1 Adversarial REINFORCE

The adversarial REINFORCE algorithm consists of two components: a generative model G and a discriminative model D.

Generative model The generative model G defines the policy that generates a response y given dialogue history x. It takes a form similar to SEQ2SEQ models, which first map the source input to a vector representation using a recurrent net and then compute the probability of generating each token in the target using a softmax function.

Discriminative model The discriminative model D is a binary classifier that takes as input a sequence of dialogue utterances $\{x, y\}$ and outputs a label indicating whether the input is generated by humans or machines. The input dialogue is encoded into a vector representation using a hierarchical encoder (Li et al., 2015; Serban et al., 2016b),[2] which is then fed to a 2-class softmax function, returning the probability of the input dialogue episode being a machine-generated dialogue (denoted $Q_-(\{x, y\})$) or a human-generated dialogue (denoted $Q_+(\{x, y\})$).

Policy Gradient Training The key idea of the system is to encourage the generator to generate utterances that are indistinguishable from human generated dialogues. We use policy gradient methods to achieve such a goal, in which the score of current utterances being human-generated ones assigned by the discriminator (i.e., $Q_+(\{x, y\})$) is used as a reward for the generator, which is trained to maximize the expected reward of generated utterance(s) using the REINFORCE algorithm (Williams, 1992):

$$J(\theta) = \mathbb{E}_{y\sim p(y|x)}(Q_+(\{x, y\})|\theta) \qquad (1)$$

Given the input dialogue history x, the bot generates a dialogue utterance y by sampling from the policy. The concatenation of the generated utterance y and the input x is fed to the discriminator. The gradient of (1) is approximated using the likelihood ratio trick (Williams, 1992; Glynn, 1990; Aleksandrov et al., 1968):

$$\begin{aligned}\nabla J(\theta) &\approx [Q_+(\{x, y\}) - b(\{x, y\})] \\ &\quad \nabla \log \pi(y|x) \\ &= [Q_+(\{x, y\}) - b(\{x, y\})] \\ &\quad \nabla \sum_t \log p(y_t|x, y_{1:t-1}) \qquad (2)\end{aligned}$$

where π denotes the probability of the generated responses. $b(\{x, y\})$ denotes the baseline value to reduce the variance of the estimate while keeping it unbiased.[3] The discriminator is simultaneously updated with the human generated dialogue that contains dialogue history x as a positive example and the machine-generated dialogue as a negative example.

3.2 Reward for Every Generation Step (REGS)

The REINFORCE algorithm described has the disadvantage that the expectation of the reward is approximated by only one sample, and the reward associated with this sample (i.e., $[Q_+(\{x, y\}) - b(\{x, y\})]$ in Eq(2)) is used for all actions (the generation of each token) in the generated sequence. Suppose, for example, the input history is *what's your name*, the human-generated response is *I am John*, and the machine-generated response is *I don't know*. The vanilla REINFORCE model assigns the same negative reward to all tokens within the human-generated response (i.e., *I*, *don't*, *know*), whereas proper credit assignment in training would give separate rewards, most likely a neutral reward for the token *I*, and negative rewards to *don't* and *know*. We call this *reward for every generation step*, abbreviated *REGS*.

Rewards for intermediate steps or partially decoded sequences are thus necessary. Unfortunately, the discriminator is trained to assign scores to fully

[1] We approximate the dialogue history using the concatenation of two preceding utterances. We found that using more than 2 context utterances yields very tiny performance improvements for SEQ2SEQ models.

[2] To be specific, each utterance p or q is mapped to a vector representation h_p or h_q using LSTM (Hochreiter and Schmidhuber, 1997). Another LSTM is put on sentence level, mapping the context dialogue sequence to a single representation.

[3] Like Ranzato et al. (2016), we train another neural network model (the critic) to estimate the value (or future reward) of current state (i.e., the dialogue history) under the current policy π. The critic network takes as input the dialogue history, transforms it to a vector representation using a hierarchical network and maps the representation to a scalar. The network is optimized based on the mean squared loss between the estimated reward and the real reward.

generated sequences, but not partially decoded ones. We propose two strategies for computing intermediate step rewards by (1) using Monte Carlo (MC) search and (2) training a discriminator that is able to assign rewards to partially decoded sequences.

In (1) Monte Carlo search, given a partially decoded s_P, the model keeps sampling tokens from the distribution until the decoding finishes. Such a process is repeated N (set to 5) times and the N generated sequences will share a common prefix s_P. These N sequences are fed to the discriminator, the average score of which is used as a reward for the s_P. A similar strategy is adopted in Yu et al. (2016a). The downside of MC is that it requires repeating the sampling process for each prefix of each sequence and is thus significantly time-consuming.[4]

In (2), we directly train a discriminator that is able to assign rewards to both fully and partially decoded sequences. We break the generated sequences into partial sequences, namely $\{y^+_{1:t}\}_{t=1}^{N_{Y^+}}$ and $\{y^-_{1:t}\}_{t=1}^{N_{Y^-}}$ and use all instances in $\{y^+_{1:t}\}_{t=1}^{N_{Y^+}}$ as positive examples and instances $\{y^-_{1:t}\}_{t=1}^{N_{Y^-}}$ as negative examples. The problem with such a strategy is that earlier actions in a sequence are shared among multiple training examples for the discriminator (for example, token y^+_1 is contained in all partially generated sequences, which results in overfitting. To mitigate this problem, we adopt a strategy similar to when training value networks in *AlphaGo* (Silver et al., 2016), in which for each collection of subsequences of Y, we randomly sample only one example from $\{y^+_{1:t}\}_{t=1}^{N_{Y^+}}$ and one example from $\{y^-_{1:t}\}_{t=1}^{N_{Y^-}}$, which are treated as positive and negative examples to update the discriminator. Compared with the Monte Carlo search model, this strategy is significantly more time-effective, but comes with the weakness that the discriminator becomes less accurate after partially decoded sequences are added in as training examples. We find that the MC model performs better when training time is less of an issue.

For each partially-generated sequence $Y_t = y_{1:t}$, the discriminator gives a classification score $Q_+(x, Y_t)$. We compute the baseline $b(x, Y_t)$ using a similar model to the vanilla REINFORCE model. This yields the following gradient to update the generator:

$$\nabla J(\theta) \approx \sum_t (Q_+(x, Y_t) - b(x, Y_t)) \nabla \log p(y_t|x, Y_{1:t-1}) \quad (3)$$

Comparing (3) with (2), we can see that the values for rewards and baselines are different among generated tokens in the same response.

Teacher Forcing Practically, we find that updating the generative model only using Eq. 1 leads to unstable training for both vanilla Reinforce and REGS, with the perplexity value skyrocketing after training the model for a few hours (even when the generator is initialized using a pre-trained SEQ2SEQ model). The reason this happens is that the generative model can only be indirectly exposed to the gold-standard target sequences through the reward passed back from the discriminator, and this reward is used to promote or discourage its (the generator's) own generated sequences. Such a training strategy is fragile: once the generator (accidentally) deteriorates in some training batches and the discriminator consequently does an extremely good job in recognizing sequences from the generator, the generator immediately gets lost. It knows that its generated sequences are bad based on the rewards outputted from the discriminator, but it does not know what sequences are good and how to push itself to generate these good sequences (the odds of generating a good response from random sampling are minute, due to the vast size of the space of possible sequences). Loss of the reward signal leads to a breakdown in the training process.

To alleviate this issue and give the generator more direct access to the gold-standard targets, we propose also feeding human generated responses to the generator for model updates. The most straightforward strategy is for the discriminator to automatically assign a reward of 1 (or other positive values) to the human generated responses and for the generator to use this reward to update itself on human generated examples. This can be seen as having a teacher intervene with the generator some fraction of the time and force it to generate the true responses, an approach that is similar to the professor-forcing algorithm of Lamb et al. (2016).

A closer look reveals that this modification is the same as the standard training of SEQ2SEQ mod-

[4] Consider one target sequence with length 20, we need to sample 5*20=100 full sequences to get rewards for all intermediate steps. Training one batch with 128 examples roughly takes roughly 1 min on a single GPU, which is computationally intractable considering the size of the dialogue data we have. We thus parallelize the sampling processes, distributing jobs across 8 GPUs.

For number of training iterations **do**
. **For** i=1,D-steps **do**
. Sample (X,Y) from real data
. Sample $\hat{Y} \sim G(\cdot|X)$
. Update D using (X, Y) as positive examples and $(X, \hat{Y})$ as negative examples.
. **End**
.
. **For** i=1,G-steps **do**
. Sample (X,Y) from real data
. Sample $\hat{Y} \sim G(\cdot|X)$
. Compute Reward r for $(X, \hat{Y})$ using D.
. Update G on $(X, \hat{Y})$ using reward r
. Teacher-Forcing: Update G on (X, Y)
. **End**
End

Figure 1: A brief review of the proposed adversarial reinforcement algorithm for training the generator G and discriminator D. The reward r from the discriminator D can be computed using different strategies according to whether using REINFORCE or REGS. The update of the generator G on $(X, \hat{Y})$ can be done by either using Eq.2 or Eq.3. D-steps is set to 5 and G-steps is set to 1.

els, making the final training alternately update the SEQ2SEQ model using the adversarial objective and the MLE objective. One can think of the professor-forcing model as a regularizer to regulate the generator once it starts deviating from the training dataset.

We also propose another workaround, in which the discriminator first assigns a reward to a human generated example using its own model, and the generator then updates itself using this reward on the human generated example only if the reward is larger than the baseline value. Such a strategy has the advantage that different weights for model updates are assigned to different human generated examples (in the form of different reward values produced by the generator) and that human generated examples are always associated with non-negative weights.

A sketch of the proposed model is shown in Figure 1.

3.3 Training Details

We first pre-train the generative model by predicting target sequences given the dialogue history. We trained a SEQ2SEQ model (Sutskever et al., 2014) with an attention mechanism (Bahdanau et al., 2015; Luong et al., 2015) on the OpenSubtitles dataset. We followed protocols recommended by Sutskever et al. (2014), such as gradient clipping, mini-batch and learning rate decay. We also pre-train the discriminator. To generate negative examples, we decode part of the training data. Half of the negative examples are generated using beam-search with mutual information reranking as described in Li et al. (2016a), and the other half is generated from sampling.

For data processing, model training and decoding (both the proposed adversarial training model and the standard SEQ2SEQ models), we employ a few strategies that improve response quality, including: (2) Remove training examples with length of responses shorter than a threshold (set to 5). We find that this significantly improves the general response quality.[5] (2) Instead of using the same learning rate for all examples, using a weighted learning rate that considers the average tf-idf score for tokens within the response. Such a strategy decreases the influence from dull and generic utterances.[6] (3) Penalizing intra-sibling ranking when doing beam search decoding to promote N-best list diversity as described in Li et al. (2016c). (4) Penalizing word types (stop words excluded) that have already been generated. Such a strategy dramatically decreases the rate of repetitive responses such as *no. no. no. no. no.* or contradictory responses such as *I don't like oranges but i like oranges*.

4 Adversarial Evaluation

In this section, we discuss strategies for successful adversarial evaluation. Note that the proposed adversarial training and adversarial evaluation are separate procedures. They are independent of each other and share no common parameters.

The idea of adversarial evaluation, first proposed by Bowman et al. (2016), is to train a discriminant function to separate generated and true sentences, in an attempt to evaluate the model's sentence generation capability. The idea has been preliminarily studied by Kannan and Vinyals (2016) in the context of dialogue generation. Adversarial evaluation also resembles the idea of the Turing test, which

[5] To compensate for the loss of short responses, one can train a separate model using short sequences.

[6] We treat each sentence as a document. Stop words are removed. Learning rates are normalized within one batch. For example, suppose t_1, t_2, ..., t_i, ... ,t_N denote the tf-idf scores for sentences within current batch and lr denotes the original learning rate. The learning rate for sentence with index i is $N \cdot lr \cdot \frac{t_i}{\sum_{i'} t_{i'}}$. To avoid exploding learning rates for sequences with extremely rare words, the tf-idf score of a sentence is capped at L times the minimum tf-idf score in the current batch. L is empirically chosen and is set to 3.

requires a human evaluator to distinguish machine-generated texts from human-generated ones. Since it is time-consuming and costly to ask a human to talk to a model and give judgements, we train a machine evaluator in place of the human evaluator to distinguish the human dialogues and machine dialogues, and we use it to measure the general quality of the generated responses.

Adversarial evaluation involves both training and testing. At training time, the evaluator is trained to label dialogues as machine-generated (negative) or human-generated (positive). At test time, the trained evaluator is evaluated on a held-out dataset. If the human-generated dialogues and machine-generated ones are indistinguishable, the model will achieve 50 percent accuracy at test time.

4.1 Adversarial Success

We define Adversarial Success (*AdverSuc* for short) to be the fraction of instances in which a model is capable of fooling the evaluator. *AdverSuc* is the difference between 1 and the accuracy achieved by the evaluator. Higher values of *AdverSuc* for a dialogue generation model are better.

4.2 Testing the Evaluator's Ability

One caveat with the adversarial evaluation methods is that they are model-dependent. We approximate the human evaluator in the Turing test with an automatic evaluator and assume that the evaluator is perfect: low accuracy of the discriminator should indicate high quality of the responses, since we interpret this to mean the generated responses are indistinguishable from the human ones. Unfortunately, there is another factor that can lead to low discriminative accuracy: a poor discriminative model. Consider a discriminator that always gives random labels or always gives the same label. Such an evaluator always yields a high *AdverSuc* value of 0.5. Bowman et al. (2016) propose two different discriminator models separately using *unigram* features and *neural* features. It is hard to tell which feature set is more reliable. The standard strategy of testing the model on a held-out development set is not suited to this case, since a model that overfits the development set is necessarily superior.

To deal with this issue, we propose setting up a few manually-invented situations to test the ability of the automatic evaluator. This is akin to setting up examinations to test the ability of the human evaluator in the Turing test. We report not only the *AdverSuc* values, but also the scores that the evaluator achieves in these manually-designed test cases, indicating how much we can trust the reported *AdverSuc*. We develop scenarios in which we know in advance how a perfect evaluator should behave, and then compare *AdverSuc* from a discriminative model with the gold-standard *AdverSuc*. Scenarios we design include:

- We use human-generated dialogues as both positive examples and negative examples. A perfect evaluator should give an *AdverSuc* of 0.5 (accuracy 50%), which is the gold-standard result.
- We use machine-generated dialogues as both positive examples and negative examples. A perfect evaluator should give an *AdverSuc* of 0.5 (accuracy 50%).
- We use original human-generated dialogues as positive examples and dialogues consisting of random utterances as negative examples. A perfect evaluator should give an *AdverSuc* of 0 (accuracy 100%).
- We use original human-generated dialogues as positive examples and dialogues with responses replaced by the utterances immediately following the true responses. A perfect evaluation should give an *AdverSuc* of 0 (accuracy 100%).

The *evaluator reliability error* (ERE) is the average deviation of an evaluator's adversarial error from the gold-standard error in the above tasks, with equal weight for each task. The smaller the error, the more reliable the evaluator is.

4.3 Machine-vs-Random Accuracy

Evaluator reliability error uses scenarios constructed from human-generated dialogues to assess feature or hyper-parameter choice for the evaluator. Unfortunately, no machine-generated responses are involved in the ERE metric. The following example illustrates the serious weakness resulting from this strategy: as will be shown in the experiment section, when inputs are decoded using greedy or beam search models, most generation systems to date yield an adversarial success less than 10 percent (evaluator accuracy 90 percent). But when using sampling for decoding, the adversarial success skyrockets to around 40 percent,[7] only 10 percent less than what's needed to pass the Turing test. A close look at the decoded sequences using sampling tells a different story: the responses from

[7] Similar results are also reported in Kannan and Vinyals (2016).

Input	tell me ... how long have you had this falling sickness ?
Vanilla-MLE	i 'm not a doctor .
Vanilla-Sample	well everything you did was totally untrue .
REINFORCE	i don 't know how long it 's been .
REGS Monte Carlo	A few months, I guess .
Input	so i had the doctors test sammy 's response to conditioning .
Vanilla-MLE	sammy wrote the test sammy wrote the test .
Vanilla-Sample	objects pick objects up objects objects objects objects objects
REINFORCE	i 'm not sure that 's relevant .
REGS Monte Carlo	so he took the pills .
Input	what are you going to do ?
Vanilla-MLE	i 'm going to the movies .
Vanilla-Sample	get him outta here first !
REINFORCE	i 'm going to get you .
REGS Monte Carlo	i 'm going to get you a new car .
Input	they fear your power your intellect .
Vanilla-MLE	you 're the only one who knows what 's going on .
Vanilla-Sample	when they are conquered and you surrender they will control all of us .
REINFORCE	i 'm afraid i 'm not ready yet .
REGS Monte Carlo	i 'm not afraid of your power .

Table 1: Sampled responses from different models. More in Appendix Tables 5 and 6.

sampling are sometimes incoherent, irrelevant or even ungrammatical.

We thus propose an additional sanity check, in which we report the accuracy of distinguishing between machine-generated responses and randomly sampled responses (*machine-vs-random* for short). This resembles the N-choose-1 metric described in Shao et al. (2017). Higher accuracy indicates that the generated responses are distinguishable from randomly sampled human responses, indicating that the generative model is not fooling the generator simply by introducing randomness. As we will show in Sec. 5, using sampling results in high *AdverSuc* values but low *machine-vs-random* accuracy.

5 Experimental Results

In this section, we detail experimental results on adversarial success and human evaluation.

Setting	ERE
SVM+Unigram	0.232
Concat Neural	0.209
Hierarchical Neural	0.193
SVM+Neural+multil-features	0.152

Table 2: ERE scores obtained by different models.

5.1 Adversarial Evaluation

ERE We first test adversarial evaluation models with different feature sets and model architectures for reliability, as measured by evaluator reliability error (ERE). We explore the following models: (1) *SVM+Unigram*: SVM using unigram features.[8] A multi-utterance dialogue (i.e., input messages and responses) is transformed to a unigram representation; (2) *Concat Neural*: a neural classification model with a softmax function that takes as input the concatenation of representations of constituent dialogues sentences; (3) *Hierarchical Neural*: a hierarchical encoder with a structure similar to the discriminator used in the reinforcement; and (4) *SVM+Neural+multi-lex-features*: a SVM model that uses the following features: unigrams, neural representations of dialogues obtained by the neural model trained using strategy (3),[9] the forward likelihood $\log p(t|s)$ and backward likelihood $p(s|t)$.

ERE scores obtained by different models are reported in Table 2. As can be seen, the *hierarchical neural* evaluator (model 3) is more reliable than simply concatenating the sentence-level representations (model 2). Using the combination of neural features and lexicalized features yields the most reliable evaluator. For the rest of this section, we report results obtained by the *Hierarchical Neural* setting due to its end-to-end nature, despite its inferiority to *SVM+Neural+multil-features*.

Table 3 presents *AdverSuc* values for different models, along with *machine-vs-random* accuracy described in Section 4.3. Higher values of *AdverSuc* and *machine-vs-random* are better.

Baselines we consider include standard SEQ2SEQ models using greedy decoding (*MLE-greedy*), beam-search (*MLE+BS*) and sampling, as well as the mutual information reranking model of Li et al. (2016a) with two algorithmic variations: (1) MMI+$p(t|s)$, in which a large N-best list is first

[8] Trained using the SVM-Light package (Joachims, 2002).

[9] The representation before the softmax layer.

Model	*AdverSuc*	*machine-vs-random*
MLE-BS	0.037	0.942
MLE-Greedy	0.049	0.945
MMI+$p(t\|s)$	0.073	0.953
MMI-$p(t)$	0.090	0.880
Sampling	0.372	0.679
Adver-Reinforce	0.080	0.945
Adver-REGS	0.098	0.952

Table 3: *AdverSuc* and *machine-vs-random* scores achieved by different training/decoding strategies.

generated using a pre-trained SEQ2SEQ model and then reranked by the backward probability $p(s|t)$ and (2) MMI$-p(t)$, in which language model probability is penalized during decoding.

Results are shown in Table 3. What first stands out is decoding using sampling (as discussed in Section 4.3), achieving a significantly higher *AdverSuc* number than all the rest models. However, this does not indicate the superiority of the sampling decoding model, since the *machine-vs-random* accuracy is at the same time significantly lower. This means that sampled responses based on SEQ2SEQ models are not only hard for an evaluator to distinguish from real human responses, but also from randomly sampled responses. A similar, though much less extreme, effect is observed for MMI$-p(t)$, which has an *AdverSuc* value slightly higher than *Adver-Reinforce*, but a significantly lower *machine-vs-random* score.

By comparing different baselines, we find that MMI+$p(t|s)$ is better than *MLE-greedy*, which is in turn better than *MLE+BS*. This result is in line with human-evaluation results from Li et al. (2016a). The two proposed adversarial algorithms achieve better performance than the baselines. We expect this to be the case, since the adversarial algorithms are trained on an objective function more similar to the evaluation metric (i.e., adversarial success). *REGS* performs slightly better than the vanilla REINFORCE algorithm.

5.2 Human Evaluation

For human evaluation, we follow protocols defined in Li et al. (2016d), employing crowdsourced judges to evaluate a random sample of 200 items. We present both an input message and the generated outputs to 3 judges and ask them to decide which of the two outputs is better (*single-turn* general quality). Ties are permitted. Identical strings are assigned the same score. We also present the judges with *multi-turn* conversations simulated between the two agents. Each conversation consists of 3 turns. Results are presented in Table 4. We observe a significant quality improvement on both single-turn quality and multi-turn quality from the proposed adversarial model. It is worth noting that the reinforcement learning system described in Li et al. (2016d), which simulates conversations between two bots and is trained based on manually designed reward functions, only improves multi-turn dialogue quality, while the model described in this paper improves both single-turn and multi-turn dialogue generation quality. This confirms that the reward adopted in adversarial training is more general, natural and effective in training dialogue systems.

Setting	adver-win	adver-lose	tie
single-turn	0.62	0.18	0.20
multi-turn	0.72	0.10	0.18

Table 4: The gain from the proposed adversarial model over the mutual information system based on pairwise human judgments.

6 Conclusion and Future Work

In this paper, drawing intuitions from the Turing test, we propose using an adversarial training approach for response generation. We cast the model in the framework of reinforcement learning and train a generator based on the signal from a discriminator to generate response sequences indistinguishable from human-generated dialogues. We observe clear performance improvements on multiple metrics from the adversarial training strategy.

The adversarial training model should theoretically benefit a variety of generation tasks in NLP. Unfortunately, in preliminary experiments applying the same training paradigm to machine translation, we did not observe a clear performance boost. We conjecture that this is because the adversarial training strategy is more beneficial to tasks in which there is a big discrepancy between the distributions of the generated sequences and the reference target sequences. In other words, the adversarial approach is more beneficial on tasks in which entropy of the targets is high. Exploring this relationship further is a focus of our future work.

Acknowledgements The authors thank Michel Galley, Bill Dolan, Chris Brockett, Jianfeng Gao and other members of the NLP group at Mi-

crosoft Research, as well as Sumit Chopra and Marc'Aurelio Ranzato from Facebook AI Research for helpful discussions and comments. Jiwei Li is supported by a Facebook Fellowship, which we gratefully acknowledge. This work is also partially supported by the NSF under award IIS-1514268, and the DARPA Communicating with Computers (CwC) program under ARO prime contract no. W911NF- 15-1-0462, IIS-1464128. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA, the NSF, or Facebook.

References

V. M. Aleksandrov, V. I. Sysoyev, and V. V. Shemeneva. 1968. Stochastic optimization. *Engineering Cybernetics* 5:11–16.

Nabiha Asghar, Pasca Poupart, Jiang Xin, and Hang Li. 2016. Online sequence-to-sequence reinforcement learning for open-domain conversational agents. *arXiv preprint arXiv:1612.03929* .

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2017. An actor-critic algorithm for sequence prediction. *ICLR* .

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proc. of ICLR*.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. *CoNLL* .

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. 2016a. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances In Neural Information Processing Systems*. pages 2172–2180.

Xilun Chen, Ben Athiwaratkun, Yu Sun, Kilian Weinberger, and Claire Cardie. 2016b. Adversarial deep averaging networks for cross-lingual sentiment classification. *arXiv preprint arXiv:1606.01614* .

Emily L Denton, Soumith Chintala, Rob Fergus, et al. 2015. Deep generative image models using a? laplacian pyramid of adversarial networks. In *Advances in neural information processing systems*. pages 1486–1494.

Peter W Glynn. 1990. Likelihood ratio gradient estimation for stochastic systems. *Communications of the ACM* 33(10):75–84.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems*. pages 2672–2680.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Thorsten Joachims. 2002. *Learning to classify text using support vector machines: Methods, theory and algorithms*. Kluwer Academic Publishers.

Anjuli Kannan and Oriol Vinyals. 2016. Adversarial evaluation of dialogue models. In *NIPS 2016 Workshop on Adversarial Training*.

Alex Lamb, Anirudh Goyal, Ying Zhang, Saizheng Zhang, Aaron Courville, and Yoshua Bengio. 2016. Professor forcing: A new algorithm for training recurrent networks. In *Advances In Neural Information Processing Systems*. pages 4601–4609.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016a. A diversity-promoting objective function for neural conversation models. In *Proc. of NAACL-HLT*.

Jiwei Li, Michel Galley, Chris Brockett, Georgios Spithourakis, Jianfeng Gao, and Bill Dolan. 2016b. A persona-based neural conversation model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany, pages 994–1003. http://www.aclweb.org/anthology/P16-1094.

Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. 2015. A hierarchical neural autoencoder for paragraphs and documents. *ACL* .

Jiwei Li, Will Monroe, and Dan Jurafsky. 2016c. A simple, fast diverse decoding algorithm for neural generation. *arXiv preprint arXiv:1611.08562* .

Jiwei Li, Will Monroe, Alan Ritter, and Dan Jurafsky. 2016d. Deep reinforcement learning for dialogue generation. *EMNLP* .

Chia-Wei Liu, Ryan Lowe, Iulian V Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How NOT to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. *EMNLP* .

Ryan Lowe, Michael Noseworthy, Iulian Serban, Nicolas Angelard-Gontier, Yoshua Bengio, and Joelle Pineau. 2017. Towards an automatic turing test: Learning to evaluate dialogue responses. *ACL* .

Ryan Lowe, Iulian V Serban, Mike Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. On the evaluation of dialogue systems with next utterance classification. *SIGDIAL* .

Yi Luan, Yangfeng Ji, and Mari Ostendorf. 2016. LSTM based conversation models. *arXiv preprint arXiv:1603.09457* .

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *ACL* .

Alec Radford, Luke Metz, and Soumith Chintala. 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434* .

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. *ICLR* .

Alan Ritter, Colin Cherry, and William B Dolan. 2011. Data-driven response generation in social media. In *Proceedings of EMNLP 2011*. pages 583–593.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. 2016. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*. pages 2226–2234.

Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016a. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of AAAI*.

Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016b. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI-16)*.

Iulian Vlad Serban, Tim Klinger, Gerald Tesauro, Kartik Talamadupula, Bowen Zhou, Yoshua Bengio, and Aaron Courville. 2016c. Multiresolution recurrent neural networks: An application to dialogue response generation. *arXiv preprint arXiv:1606.00776* .

Iulian Vlad Serban, Ryan Lowe, Laurent Charlin, and Joelle Pineau. 2016d. Generative deep neural networks for dialogue: A short review .

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. *AAAI* .

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *Proceedings of ACL-IJCNLP*. pages 1577–1586.

Louis Shao, Stephan Gouws, Denny Britz, Anna Goldie, Brian Strope, and Ray Kurzweil. 2017. Generating long and diverse responses with neural conversational models. *ICLR* .

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Minimum risk training for neural machine translation. *ACL* .

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature* 529(7587):484–489.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Meg Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. In *Proceedings of NAACL-HLT*.

Pei-Hao Su, Milica Gasic, Nikola Mrksic, Lina Rojas-Barahona, Stefan Ultes, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Continuously learning neural dialogue management. *arxiv* .

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. pages 3104–3112.

Alan M Turing. 1950. Computing machinery and intelligence. *Mind* 59(236):433–460.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. In *Proceedings of ICML Deep Learning Workshop*.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Lina M Rojas-Barahona, Pei-Hao Su, Stefan Ultes, David Vandyke, and Steve Young. 2016. A network-based end-to-end trainable task-oriented dialogue system. *arXiv preprint arXiv:1604.04562* .

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8(3-4):229–256.

Sam Wiseman and Alexander M Rush. 2016. Sequence-to-sequence learning as beam-search optimization. *ACL* .

Zhen Xu, Bingquan Liu, Baoxun Wang, Chengjie Sun, and Xiaolong Wang. 2016. Incorporating loose-structured knowledge into LSTM with recall gate for conversation modeling. *arXiv preprint arXiv:1605.05110* .

Kaisheng Yao, Geoffrey Zweig, and Baolin Peng. 2015. Attention with intention for a neural network conversation model. In *NIPS workshop on Machine Learning for Spoken Language Understanding and Interaction*.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2016a. Seqgan: sequence generative adversarial nets with policy gradient. *arXiv preprint arXiv:1609.05473* .

Zhou Yu, Ziyu Xu, Alan W Black, and Alex I Rudnicky. 2016b. Strategy and policy learning for non-task-oriented conversational systems. In *17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. page 404.

Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. 2017. Aspect-augmented adversarial networks for domain adaptation. *arXiv preprint arXiv:1701.00188* .

Input	maybe his eyes were too big for his stomach .
Vanilla-MLE	he 's got a big mouth .
Vanilla-Sample	i 'm sure he 's unhappy with the fact .
Reinforce	he 's a little bit of a pain .
MC	i don 't think so .
Input	they didn 't have much success with this problem commander .
Vanilla-MLE	they 're not the only ones who have been in the system .
Vanilla-Sample	it 's the new priority only .
Reinforce	they 're not the only ones who can help us .
MC	can we find someone else ?
Input	and he carried them all the time wherever he went .
Vanilla-MLE	i don 't know who he is .
Vanilla-Sample	the church ... suffered the winds of 900 meters with horse aid .
Reinforce	he was a good man .
MC	he was a very good man .
Input	somebody probably flushed their pet snake down the toilet .
Vanilla-MLE	i 'm sure the dog will be here soon .
Vanilla-Sample	i could at least think .
Reinforce	i don 't think so .
MC	i don 't think so .
Input	excuse me but how did we pay for these suits ?
Vanilla-MLE	i 'm not a beggar .
Vanilla-Sample	is this the money eugene ?
Reinforce	i don 't know what to do .
MC	how much money you owe me ?
Input	... while the beautiful and happy world was madly destroyed ;
Vanilla-MLE	i was born in the world of the world
Vanilla-Sample	as in childhood i took you to one tenth of a postcard
Reinforce	and the world was full of joy .
MC	and the world was full of joy .
Input	i was kinda hoping that you were my dad .
Vanilla-MLE	i 'm just a little bit of a little bit of a little bit of a little bit of a
Vanilla-Sample	shiny gold gold and silver in the mountains .
Reinforce	i 'm not gonna be a baby .
MC	i 'm not going to be a father .
Input	the man who does that is not afraid to die .
Vanilla-MLE	he 's not afraid of the dark .
Vanilla-Sample	even the program is the only truth
Reinforce	he 's also not afraid of me .
MC	he 's such a brave man !
Input	do you have any idea what caused the explosion ?
Vanilla-MLE	i think it 's the same thing .
Vanilla-Sample	i 'm still very nervous .
Reinforce	maybe it 's just a coincidence .
MC	i don 't know if it 's a coincidence .
Input	don 't bother the citizens in hong kong island anymore
Vanilla-MLE	i 'll be back in a few days
Vanilla-Sample	you wiii be transferred to the target court
Reinforce	i 'm not sure if you can handle it without bothering them .
MC	i 'm not going to be a nuisance
Input	you know you ought to find yourself a girl .
Vanilla-MLE	you know what i mean ?
Vanilla-Sample	it 's full of crime ...
Reinforce	i don 't know what you 're talking about .
MC	i don 't want to be a girl .
Input	i got you the big screen tv deluxe UNknown machine
Vanilla-MLE	i 'm gonna be a little late for the movie .
Vanilla-Sample	you mean you 're living ?
Reinforce	i 'm gonna get you a new one .
MC	i 'm gonna get you a new tv set .
Input	we won 't give her the chance to surprise us .
Vanilla-MLE	she 's not going to be a party member .
Vanilla-Sample	the timing 's good and bad .
Reinforce	we 'll have to get her back .
MC	i don 't think we have to worry about her .
Input	does it live in this matter we call dna ?
Vanilla-MLE	i don 't think so .
Vanilla-Sample	and it requires time to be wrong and no one believes it .
Reinforce	it 's a genetic disorder .
MC	i don 't think so .

Table 5: Appendix: Responses sampled from different models.

Input	i 'll have to drive home .
Vanilla-MLE	i 'll be back in a minute .
Vanilla-Sample	let 's go for some fun .
Reinforce	i 'll drive you home .
MC	i don 't have a car .
Input	i had to work late and then i went for a walk .
Vanilla-MLE	i was in the middle of the road .
Vanilla-Sample	my body was just pounding .
Reinforce	i was just trying to help you .
MC	i was just trying to get a little sleep .
Input	lots of cream lots of sugar .
Vanilla-MLE	i 'm gonna be in the kitchen .
Vanilla-Sample	was this gay or red ?
Reinforce	i don 't want to be a vegetable .
MC	i 'm not a very good cook .
Input	so they just let him go ?
Vanilla-MLE	he 's not gonna let them go through the motions .
Vanilla-Sample	they asked him whether we could go see ernesto ?
Reinforce	i don 't know what to do .
MC	he 's not going to be here any longer .
Input	you must not bomb this town .
Vanilla-MLE	you have to go back to the city .
Vanilla-Sample	the one is a multiple choice .
Reinforce	I think I have to .
MC	I think I have to .
Input	communism recently failed in eastern europe .
Vanilla-MLE	the war is not over .
Vanilla-Sample	there are people who hope that in tokyo they won 't risk losing their departing .
Reinforce	and the people who are not the same .
MC	the war is still going on .
Input	you should stop in more often .
Vanilla-MLE	I don 't think so .
Vanilla-Sample	is it possible as old as you ?
Reinforce	i 'm not going to be a burden on you .
MC	i don 't want to be a burden .
Input	he couldn 't have heard us .
Vanilla-MLE	he was in the middle of the night .
Vanilla-Sample	he should have been somewhere out here .
Reinforce	he 's not a good guy .
MC	he 's got a lot of nerve.
Input	that 's our wake up call .
Vanilla-MLE	we 're gonna be late for the meeting .
Vanilla-Sample	c 'mon c 'mon let 's get outta here .
Reinforce	i 'll be right back .
MC	we 're gonna have to get to the station .
Input	i had 'em when i started .
Vanilla-MLE	i 'm gonna get 'em .
Vanilla-Sample	those weak rotten guys had me in a rush on a broken swoop .
Reinforce	i don 't know what to do .
MC	i was just a little bit nervous .
Input	oatmeal delicious start into a new day
Vanilla-MLE	i 'll be right back
Vanilla-Sample	sure if you don 't put it into the water
Reinforce	i 'm gonna be a little busy with the dishes .
MC	i 'm gonna make you a little dinner .

Table 6: Appendix: More responses sampled from different models.