Introducing DPAC-GAN for improving Response Diversity in Dialogue Generation

Joris Baan
University of Amsterdam
Amsterdam, The Netherlands

Petra Ormel
University of Amsterdam
Amsterdam, The Netherlands

Laurens Samson
University of Amsterdam
Amsterdam, The Netherlands

Jarno Verhagen
University of Amsterdam
Amsterdam, The Netherlands

ABSTRACT

A problem with current dialogue generation algorithms is that they tend to produce uninformative replies like "I don't know" or "I'm sorry". Xu et al. [21] proposed DP-GAN to promote diversity in text generation through generative adversarial training and the use of a language model based discriminator. Current work suggests an actor-critic [17] architecture that builds upon DP-GAN to improve the quality and diversity of replies in dialogue generation. However, preliminary results from DPAC-GAN are inconclusive and more elaborate testing is required. Still, this work might pave the way for future research on diversity in dialogue generation. Additionaly, SeqGAN and DP-GAN are implemented and tested on the Daily Dialogue dataset to reproduce and compare the original results [8, 21].

1 INTRODUCTION

Dialogue generation is a Natural Language Processing (NLP) task giving rise to some unique challenges. The goal is similar to other text generation tasks like machine translation and text summarization: a sequence of words is generated given another sequence of words as input. In the case of dialogue generation, the input sequence is a history of previous utterances, while the output sequence should be a suitable response. The responses not only need to be fitting, but also informative and diverse, such that the dialogue makes a human-like impression. A recurring problem with existing dialogue generation models is that generated replies tend to be boring and uninformative, e.g. "I don't know". Answers are grammatically and contextually a correct continuation of the dialogue, but they do not add any information to the dialogue. As such, they do not give the impression of a human-like response.

In recent work by Xu et al. [21], a new method is proposed for improving dialogue diversity. The proposed Diversity-Promoting Generative Adversarial Network (DP-GAN) uses a GAN to simultaneously train models both for generating replies as well as judging the quality of the generated replies. This is done by generating replies using a standard sequence to sequence (seq2seq) model, and using a language model based discriminator to give a reward for the quality of the generated reply. The generator is then trained based on the reward using standard REINFORCE Policy Gradient (Williams [19]).

A limitation of the proposed DP-GAN model is the rate of learning and quality of the estimated return. In DP-GAN a Monte-Carlo approach (REINFORCE Williams [19]) is utilized for training the generator, which performs a single update after generating a complete sentence. We propose a novel architecture called DPAC-GAN, based on the actor-critic framework [5], that updates the parameters after every generated word. DPAC-GAN introduces a critic network that predicts the expected return of an intermediate sentence, which should make DPAC-GAN better able to predict the long-term reward of an intermediate generated sentence. Additionally, we implement and test SeqGAN and DP-GAN on the Dialy Dialogue dataset to reproduce results and compare the two models.

2 RELATED WORK

This section aims to provide a theoretical framework for the current state-of-the-art in dialogue generation as well as work in sequence generation tasks that uses actor-critic methods.

2.1 Vanilla Seq2seq

Early neural based methods (e.g. Sutskever et al. [16]) were comprised of sequence-to-sequence models trained with maximum likelihood estimation (MLE). The history of utterances is encoded into a hidden representation through a recurrent neural network (RNN) called the encoder. The hidden representation is then decoded by a second RNN, called the decoder, into the next reply. The model is trained in a supervised way, using the real human reply as the target. Results are good in the sense that they are often grammatically correct, but they tend to be repetitive and uninformative.

2.2 Reinforcement Learning Seq2seq

Li et al. [7] attempted to improve on this method by applying reinforcement learning instead of MLE to train the generator model. The history of the dialogue is treated like the state and the next response is treated like the action. A similar sequence-to-sequence model takes the role of policy, while the reward is calculated using manually engineered functions. Part of these reward functions is based on manually curated lists of dull responses. To improve informativeness of responses, the long-term reward is maximized. This is done by letting two agents based on the trained model continue a dialog for extended times. The reasoning behind this intuition is that it is more difficult to keep a dialog from becoming repetitive if new answers do not add information to the conversation. Results show that responses become more diverse and interactive.

2.3 SeqGAN

Another method, SeqGAN [22], extends the reinforcement learning approach by the use of Generative Adversarial Networks (GANs) that make use of a discriminator to define the reward for text generated by a generator. In GANs [3], the discriminator is trained to classify data as either real or fake, while the generator tries to fool the discriminator by generating human like data. SeqGAN uses an RNN for the generative model and a convolutional neural network (CNN) for the discriminative sequence classifier [4]. The task of the discriminator is to properly classify real-world text and generated text as human or machine created. This method was inspired by the Turing test. In SeqGAN the probability of the sentence being real is used as the reward. To circumvent the problem that the discriminator can only classify finished sentences, policy gradient updates are performed using Monte Carlo (MC) search to estimate the expected reward for an intermediate state-action pair. Experiments using metrics that capture diversity show significant improvements over baselines.

2.4 Diversity Promoting GAN

Xu et al. [21] proposed a language-model based discriminator to promote the generation of high novelty text. Instead of classifying the generator output as real or fake, the cross entropy of the output of the language-model is used as reward. Most classifier based discriminators [6, 22], take the probability of a sentence being real or fake as the reward. When generated sentences approach the real world text, the reward saturates and can barely differentiate the novelty of sentences. This is called the saturation problem and is addressed by DP-GAN by using a cross-entropy based reward for individual actions.

Xu et al. [21] assume real world text is diverse and fluent, whereas generated text is iterative or of low-quality. This should be easily distinguishable by the discriminator. To promote real world sentence generation, sentences that fit the real world text distribution are considered informative and fluent, and their rewards are maximized. Conversely, the rewards for fake, machine generated sentences are minimized. Furthermore, through the use of a language model discriminator, a world level reward can be calculated without the computational expensive Monte Carlo (MC) search. MC search samples multiple possible finished sentences from every newly generated word to estimate the reward per word in order to apply policy gradient. On human-evaluated experiments, DP-GAN achieved highest scores on relevance and fluency for the task of Dialogue Generation compared to MLE, PG-BLUE and SeqGAN.

2.5 Actor-Critic for Sequence Prediction

Recent work into the sequence generation task that involves the actor-critic approach taken in this paper is from Bahdanau et al. [1]. They used an additional critic network inspired by Reinforcement Learning literature [17] to estimate the expected return of each output token, also called the value function. The goal is to approximate the task-specific score such as BLUE [14] or ROUGE [11]. Current work wants to apply this critic network to a GAN setting where the critic is used to estimate the reward that the discriminator network would output by sampling the next action. The usage of an additional critic in the work of Bahdanau et al. [1] demonstrated improvement in synthetic tasks and machine translation [1]. This indicates that applying the actor-critic framework in GAN based dialogue generation might be prosperous as well.

3 DIVERSITY-PROMOTING ACTOR-CRITIC GAN

DPAC-GAN casts the REINFORCE Policy Gradient setting to an actor-critic Policy Gradient setting. This impacts the original DP-GAN architecture in three ways. First, an additional critic network is introduced to estimate the return G_t at each time step. In contrast, the original DP-GAN estimates the expected return G_t only *after* generating a complete reply by summing up all rewards following time step t. Second, the parameters of the actor (generator) and critic are updated at each time step, as opposed to a single update after a complete MC rollout to finish the reply. This allows for faster learning and less variance. Third, experience replay is used to decorrelate the data and make it more independent and identically distributed (iid). This is a common trick introduced by Google Deepmind [13] to stabilize training and make more efficient use of data.

3.1 Actor

The term actor is interchangeably used with generator and policy and refers to the model that actually generates the replies. The actor in DPAC-GAN is almost identical to the generator in DP-GAN and has a standard SEQ2SEQ architecture with a GRU encoder and decoder with attention [2]. Beamsearch is used during evaluation time to efficiently keep track of the top k sample replies and pick the most likely one.

The goal of the actor is to generate relevant, diverse and informative replies given a dialogue history. This is incentivized by maximizing a performance measure $J(\theta)$. In gradient descent based optimizers this results in an update rule that takes a step in the negative direction of the gradient of $J(\theta)$. The actor gradient is similar to the REINFORCE gradient which is defined in Eq 1.

$$\nabla J(\theta) = \mathbb{E}_\pi[G_t log \nabla G_\theta(A_t|S_t)] \tag{1}$$

However, instead of using episode samples to estimate the expected return $\mathbb{E}_\pi[G_t]$, a critic network is used to estimate $\mathbb{E}_\pi[G_t]$. Thus, in actor loss the log gradient of the policy is scaled with the Temporal Difference (TD) error δ, as defined in section 3.2. This results in the following update rule for the actor's parameters θ, where action A_t is sampled from the actor given the dialogue history and the current intermediate reply. Intuitively this update should increase the probability of actions that are expected to produce a high return, scaled by their probability of occurring [17].

$$\theta_{t+1} \longleftarrow \theta_t + \alpha_\theta \delta_t \nabla G_\theta(A_t|S_t) \tag{2}$$

3.2 Critic

The critic should converge to the state-action value function Q(s,a) and is similar to the proposed Deep Q-Network (DQN) from Mnih et al [13]. It consists of a GRU based sequence model that takes as input a sequence of words (state) and outputs the estimated return for every possible action. The critic loss is defined as the TD error δ (Eq 3). This loss causes the critic to minimize the error between the estimated return of the current time step and the estimated return of the best action for the next time step plus the immediate reward [17]. The critic is thus expected to learn the long-term value of a state based on the rewards given by the discriminator.

$$\delta_t = (R_{t+1} + \arg\max_a Q(S_{t+1}, a) - Q(S_t, A_t))^2 \tag{3}$$

$$w_{t+1} \leftarrow w_t + \alpha_w \delta_t \nabla Q(S_t, A_t) \tag{4}$$

A separate target network is used to estimate the value of state in the next time step, which was also introduced by Mnih et al. [13]. This additional network addresses the 'chasing your own tail' problem where the estimates of both Q terms in Eq 3 are correlated, and freezes the parameters of the target network for a number of training iterations to stabilize training.

3.3 Discriminator

In the context of the actor-critic framework, the discriminator defines the reward function. Xu et al. [21] proposed a language-model based discriminator D_ϕ. Instead of classifying the generator output as real or fake like SeqGAN, they used the cross entropy of the output of the language-model as reward. This work uses a language-model based discriminator as well, it builds upon a unidirectional GRU that takes the first 0 - t words of the reply as input and outputs the probability distribution of the successive word at t+1 over the vocabulary. In DP-GAN [21], they defined the word level reward as:

$$R(y_{t,k}) = -\log D_\phi(y_{t,k}|y_{t,<k}) \tag{5}$$

Where $y_t = (y_{t,1}, ..., y_{t,K})$ is the t^{th} reply and $y_{t,K}$ is the K^{th} word. They assume the real-world text is diverse and fluent and want to maximize the reward for sentences that look like the real-world data. However, equation 5 would give a low reward for sentences that look like real-world data. Therefore, we think the formula contains a mistake and the reward should be:

$$R(y_{t,k}) = \log D_\phi(y_{t,k}|y_{t,<k}) \tag{6}$$

The sentence-level reward for a sentence y of K words is then defined as the average reward over each word:

$$R(y_t) = \frac{1}{K}\sum_{k=1}^{K} \log D_\phi(y_{t,k}|y_{t,<k}) \tag{7}$$

The loss function stays the same as in DP-GAN and is defined as:

$$J(\phi) = -(\mathbb{E}_{y \sim p_{data}}[(R(y))] - \mathbb{E}_{y \sim G_\phi}[R(y)]) \tag{8}$$

3.4 Experience Replay

For each action the actor samples, a tuple of *(state, action, reward, next state, terminated)* is inserted into the replay memory. Instead of updating the actor on successive and correlated samples, random batches of observations are sampled from the replay memory at each time step and used to update the actor. Since the critic is an GRU based sequence model, it can handle the variable length states within a sampled batch of observations.

One problem with using experience replay is that the data used to update the policy is produced by an older version of the policy, which makes the approach off-policy. One common way to address this is to use importance sampling. However, as this makes the process messy and more computationally intensive, this work assumes that the old policy is close enough to the current policy to avoid the need for importance sampling. To further aid this assumption, the replay memory size is kept small such that the difference in the policies is kept to a minimum. Another recent paper that shows that off-policy policy gradient is possible without using importance sampling is work by Lillicrap et al. [10] on Deep Deterministic Policy Gradient. However, as they use deterministic policies, it is not entirely comparable.

3.5 Actor-Critic training

Before applying the adversarial actor-critic algorithm, the generator and discriminator are pre-trained separately. The generator is pre-trained using maximum likelihood estimation to reproduce the human written replies in the dataset, following the work of Sutskever et al. [16]. Pre-training of the discriminator is performed by maximizing its reward for real replies from the dataset and minimizing its reward for replies generated by the pre-trained generator.

Algorithm 1 describes the complete training process. The adversarial training loop comes down to updating the actor on each generated word using policy gradient, after which teacher forcing is applied to expose the actor to the gold standard. This is inspired by Li et al. [8] and should stabilize training. After repeating this procedure for the actor M times, the discriminator is trained K times.

4 EXPERIMENTS

Experiments are conducted on the Daily Dialog (Li et al. [9]) dataset. It consists of human written conversations between two persons, following bi-turn dialog flow patterns. It covers ten categories ranging from ordinary life to financial topics [9]. To simplify the task, some pre-processing steps are performed to prepare the dataset used for training. All data is lowercased and the vocabulary is limited to 8000 tokens. To limit the vocabulary, the least occurring tokens are replaced by a special *unknown* word token. Furthermore, a *start-of-sentence* and *end-of-sentence* token are added to the start and end of each sentence respectively. An *end-of-utterance* token is placed at the end of each utterance.

Each sequence of three utterances is split into a context and reply, where the context consists of the first two utterances. The task is simplified by only picking context-reply pairs of which the length of the reply is between five and twenty tokens. After pre-processing, the dataset on which the models are trained is comprised of 45.481 context-reply pairs.

At evaluation time, beam search is used to sample results from the output distribution over the vocabulary. Beam search executes a search algorithm that discards non promising sample replies based

```
Initialize $G_\theta$ and $D_\phi$ with random weights $\theta, \phi$
Pre-train $G_\theta$ using vanilla SEQ2SEQ MLE
Pre-train $D_\phi$ using samples from $G_\theta$
N = number of training iterations
M = number of training generator
K = number of training discriminator
T = maximum reply length
for each n in 1:N do
    for each m in 1:M do
        Sample context $C$ and reply $Y_{1:T} \sim Data$
        for each t in 1:T-1 do
            Sample word $w_{t+1} \sim G_\theta(context, w_{1:t})$
            Compute reward $r_{t+1}\ logD_\phi(w_{t+1}|w_{1:t})$
            Add $(w_{1:t}, w_t, r, w_{1:t+1})$ to replay memory
            Sample batch of observations from replay memory
            Predict $G_t$ using critic $V_w$
            Update actor via Eq(2) and critic via Eq(4)
        end
        Update generator on $Y_{1:T}$ via teacher forcing
    end
    for each k in 1:K do
        Generate samples using $G_\theta$ and $Data$
        Update discriminator using the loss in Eq(8)
    end
end
```
Algorithm 1: The adversarial actor-critic algorithm for training the generator G_θ, discriminator D_ϕ and critic V_w

on the combined probability of the tokens in a reply. The size of the *beam*, the remaining search alternatives, is limited by a specified beam size [23].

4.1 Metrics

Quantitative evaluation of the replies is done using various metrics. The amount of distinct n-grams (Dist-n) and replies (Dist-R) is used as an indication of the degree of diversity in the replies, similar to the evaluation of DP-GAN [21]. N-grams of sizes one, two and three are used. The total amount of generated tokens is also measured to provide more insight into the scale of generated text, and thus the relevancy of the distinct metrics. To obtain an indication whether the generated replies are on-topic given the context, the average embedding cosine similarity between the embeddings of the generated replies and the real replies is used. Embedding-based metrics can be interpreted as calculating the topicality of a proposed response [12]. Furthermore, the BLEU score based on 2-grams is used to get an idea about the linguistic quality of generated text. Research by Liu et al. [12] has shown that BLEU-3 and BLEU-4 behave as a scaled, noisy version of BLEU-2; thus if one is to evaluate dialogue responses with BLEU, they recommend the choice of N = 2 over N = 3 or 4 for evaluating dialogue responses. It is commonly agreed upon, however, that BLEU is far from a perfect score for dialogue generation tasks. Evaluation is done using NLGEval[1].

[1]Github, Maluuba/nlg-eval: https://github.com/Maluuba/nlg-eval

4.2 Baselines

For comparison with DPAC-GAN, the following models are used as a baseline:

MLE Basic sequence-to-sequence model trained with maximum likelihood estimation. Decoding is done using an attention model [2]. The MLE model used in our experiments is the same model as the pre-trained model used as basis for SeqGAN, DP-GAN and DPAC-GAN. Implementation is based on pytorch-seq2seq provided by IBM[2].

SeqGAN GAN based model in which the generator model is equal to the MLE model. Sequence GAN (Yu et al. [22]) adds a discriminator in the form of a binary classifier to simulate the Turing test. After pre-training the generator using MLE, it is further trained using adversarial learning. The classification of the discriminator holds as the reward function.

DP-GAN Model serving as a basis of DPAC-GAN. The main difference is that Diversity Promoting GAN (Xu et al. [21]) estimates G_t after generating a complete sentence, while DPAC-GAN estimates G_t already during intermediate lengths of the sentence by the use of the critic network.

4.3 Training Details

MLE pre-training of the generator is done using the Adam optimizer with a learning rate of $1 * 10^{-3}$ using a batch size of 64. Teacher forcing [20] and Dropout are used during training. Dropout is used to reduce overfitting [15]. The embedding size as well as the hidden size of the encoder in the generator is set to 256. The discriminator is pre-trained using the Adam optimizer with a learning rate of $1 * 10^{-2}$. The encoder of the discriminator has its embedding and hidden size set to 128.

For the training of Seq-GAN and DP-GAN, the Adagrad optimizer is utilized for optimizing the actor and the discriminator as suggested by Xu et al.[21]. Both the learning rates are set to $1 * 10^{-2}$. The architectures of both networks remain the same as in the pre-training phase. Furthermore, N,M,K and T are set to 3500, 1, 5 and 20 respectively. This causes the ratio of generator and discriminator updates to be 1:5 and enforces a maximum sentence length of twenty tokens. Lastly, the number of samples used for Monte-Carlo in Seq-GAN is set to 3.

5 RESULTS

5.1 Pre-training generator

The performance of the MLE generator measured on different distinct N-grams is shown in Figure 1. It can be observed that variation in replies grows during the training process.

In Figure 2 the scores on BLUE(2) and Embedding Average Cosine Similarity (EACS) are plotted during the training process. It can be observed that both scores significantly rise in the first epochs, thereafter they only slightly improve over the rest of the training. Since there are no scores of these metrics on the same dataset for comparison, it is hard to define the actual performance from

[2]Github, IBM/seq2seq: https://github.com/IBM/pytorch-seq2seq

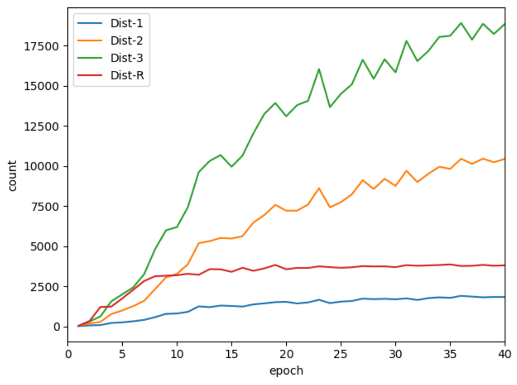

Figure 1: Count of distinct N-grams during pre-training of generator

these scores. However, it can be concluded the topicality and the linguistic quality improve during the training process.

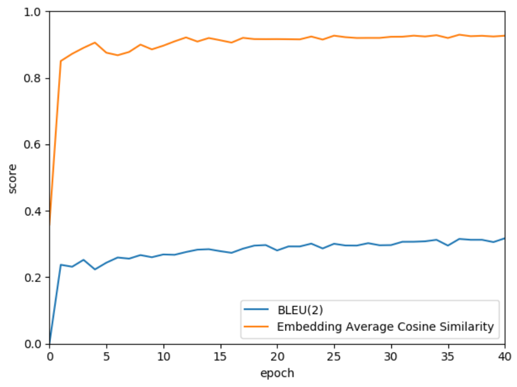

Figure 2: Scores on BLUE(2) and EACS during pre-training of generator

5.2 Pre-training discriminator

The discriminator is pre-trained on the Daily Dialog data and generated replies from the pre-trained generator. The goal of the discriminator was to maximize the reward from text that came from the dataset and to minimize the reward from text that was generated by the generator. In Figure 3 the results are shown. To smooth the graph, the average over 20 iterations is taken over a total of 3 epochs. It can be observed that the discriminator is able to learn to differentiate real-world replies from generated/fake replies reasonably fast.

5.3 SeqGAN

Quantitative results for SeqGAN after 12 epochs are shown on the left side in Table 1. The generator of SeqGAN was pre-trained for 40 epochs. The results of the pre-trained generator are shown in Figure 1 and 2. From Table 1 it can be observed that all distinctiveness metrics are reduced. This shows a reduction in the amount of word

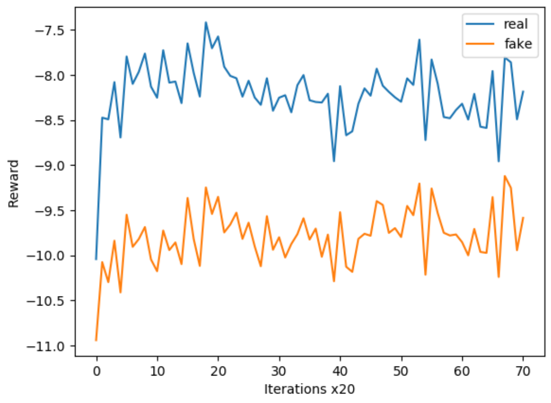

Figure 3: Rewards for real replies and generated replies during pre-training discriminator

combinations generated by the model and is an indication of lower diversity among the generated replies. The vocabulary used by the generator (indicated by the Dist-1 metric) decreased in size by fifty tokens. The total amount of generated tokens is reduced by 1131, meaning that the generated replies became shorter on average. BLEU(2) score increased with 7.3%, giving the impressions that the generated replies are more similar to the language used in the replies generated by humans. Average cosine similarity of the word embeddings increased by 1%.

5.4 DP-GAN

Quantitative result of DP-GAN after 12 epochs are shown on the right side in Table 1. For DP-GAN, the generator was pre-trained for 20 epochs instead of 40 due to limited time and computational resources. Although the difference is expected to be negligible, it could be unfair to directly compare results from SeqGAN to results from DP-GAN. DP-GAN shows a big increase in the size of the vocabulary being used in generated replies. 224 new words were used in the generated replies. Apart from the increase in vocabulary, the metrics related to distinctiveness decreased in value, indicating that diversity did not improve quantitatively. The amount of distinct replies being generated after training DP-GAN is reduced by 382. Average cosine similarity of the word embeddings increased by 1.5%. BLEU(2) is increased by 1.4% using DP-GAN.

5.5 DPAC-GAN

Preliminary results from tests with DPAC-GAN are shown in Table 2. It is important to note that these results are based on only one epoch of actor-critic training due to time and computational constraints. These preliminary results seem to indicate that the diversity in replies is increasing, but a lot of further testing is required before any real conclusions can be drawn.

	SeqGAN			DP-GAN		
	Base	Epoch 12	+/-	Base	Epoch 12	+/-
Tokens	49549	48418	-2.3%	43092	41707	-3.2%
Dist-1	1824	1774	-2.7%	1523	1747	+14.7%
Dist-2	10431	9481	-9.1%	7201	7163	-0.5%
Dist-3	18842	17656	-6.3%	13089	12285	-6.1%
Dist-R	3798	3719	-3.1%	3552	3170	-10.8%
BLEU(2)	0.317	0.340	+7.3%	0.281	0.285	+1.4%
EACS	0.927	0.936	+1.0%	0.917	0.931	+1.5%

Table 1: Quantitative results for SeqGAN and DP-GAN with respect to the pre-trained generator. EACS for Embedding Average Cosine Simlarity

	DPAC-GAN		
	Base	Epoch 1	+/-
Token	43092	46895	+8.8%
Dist-1	1523	1622	+6.5%
Dist-2	7201	7810	+8.5%
Dist-3	13089	14555	+11.2%
Dist-R	3552	3476	-3.1%
Bleu (2)	0.281	0.317	+12.8%
EACS	0.917	0.933	+1.7%

Table 2: Quantitative results for DPAC-GAN with respect to the pre-trained generator. EACS for Embedding Average Cosine Simlarity

5.6 Qualitative analysis

Although, a qualitative analysis can be tricky due to cherry picking, we believe it is a valuable addition to the quantitative measures, since they are not able to capture diversity and topicality properly.

5.7 On the Quality of Diversity Metrics

In Table 1 the quantitative results are depicted for Seq-GAN and DP-GAN. These results do not match the expectations of the models, namely that they would promote the diversity of the answers. For both models, a decrease in all diversity metrics is measured, with the exception that DP-GAN is more diverse in the use of single words (Dist-1). To see what caused this drop in the diversity metrics an additional experiment was performed that indicates the diversity of the answers does, in fact, not decrease. The set of three-grams present in the replies from the MLE Generator on the evaluation set was compared to the set of three-grams present in the output of DP-GAN.

It became apparent that the set from the MLE generator contained mostly grammatically incorrect three grams and had a size of 10425. Examples from this set are: *'play the the', 'what much is', 'I he .'*. These three-grams should intuitively not be counted for measuring diversity. After training DP-GAN, 9682 new three-grams are added to the set that were originally not in the MLE generator set. These three-grams are actually mostly grammatically correct and contribute to more diverse answers. Examples from this set are: *'I have been', 'doesn't matter.', 'you messed up'*. Although, one cannot conclude that the DP-GAN is actually providing more diverse answers, this experiment does show that these diversity measures are not actually properly measuring diversity. Note that this phenomenon occurs for all the distinct measures.

Input:
Speaker 1: *"That sounds nice and cheap."*
Speaker 2: *"Yes, you're saving money because they're on sale today."*
Real Reply: *Okay, give me the roses*
MLE Reply: *I you I'll a that wine*
Seq-GAN Reply: *I think, I'll try it on.*
DP-GAN Reply: *Okay, I'll take it*
Input:
Speaker 1: *"May I know your occupation?"*
Speaker 2: *"I'm a professor at a university"*
Real Reply: *Is that right? Which university if you don't mind my asking*
MLE Reply: *That's right. Which university do you to in?*
Seq-GAN Reply: *Do you want to put it in your passport?*
DP-GAN Reply: *Where is your problem? Which university do you want?*
Input:
Speaker 1: *"I'd like the double cheese burger meal please"*
Speaker 2: *"What kind of drink would you like with that?"*
Real Reply: *Just coke will be fine*
MLE Reply: *I like a salad with a apple salad*
Seq-GAN Reply: *Those salad a lot of apple please*
DP-GAN Reply: *I'd like a chocolate coke*
Input:
Speaker 1: *"What park do you want to go to?*
Speaker 2: *"Daniel wants to go the <unk>. While mom wants to visit the summer palace"*
Real Reply: *Haven't we been these places already?*
MLE Reply: *I guess you can do climbing that.*
Seq-GAN Reply: *So what do you want to go?*
DP-GAN Reply: *How about some beer?*

Table 3: Example replies from the different models (Real Reply, MLE approach, Seq-GAN, DP-GAN) with as input history of two utterances

Table 3 shows some generated replies from the trained models. It is remarkable how the grammar is improving in the Seq-GAN and DP-GAN replies compared to the pre-trained (MLE) model. It also seems like the content of the DP-GAN often appears to be off topic. This might be caused by the policy gradient step which tries to maximize the reward received from the language model discriminator. In Seq-GAN, the discriminator learns a reward function based on the content and reply, whereas in DP-GAN the reward function is solely based on whether the replies are good sentences. Thus, In DP-GAN, the discriminator only provides rewards for the novelty of the text, but not for the content of the answer. Nevertheless, the Seq-GAN and DP-GAN answers tend to be less boring and more informative, which is the primary goal of their architecture and training methodology.

6 DISCUSSION

The goal of this research was to encourage the generator to produce more diverse answers, while having meaningful content depending on the history. First of all, measuring diversity of language is a difficult task, most of the measures that exist do not capture diversity, which is also discussed in section 5.7. Moreover, metrics that evaluate the content and diversity of the answers at the same time are not existing. Human evaluations would have been most appropriate to capture the diversity and the content at the same time. Unfortunately, no resources were available to realize this evaluation.

The qualitative results of the diversity-promoting GAN are promising on the behalf of diversity. However, the model appears to attach too little value to the content of its answers. Ideally, one would have a model that provides informative and diverse answers. The lack of content of the DP-GAN might be caused by the absence of knowledge about the context of the discriminator. The discriminator's goal is to maximize the log likelihood of data and minimize the log likelihood of the generated data independent of the context. Future work could investigate a discriminator where the log likelihood of the real data is maximized and log likelihood of the fake data is minimized, also given on the context.

Preliminary results for DPAC-GAN are inconclusive, and the evaluation of the model requires more testing. However, the theoretical advantages are clear: less variance, intra-episode learning and a better estimate of the return using a critic network. However, as GAN-based systems on NLP tasks are already unstable, adding another bootstrapping component might make the system even more unstable. To address this instability, recent off-policy methods such as ACER, introduced by Wang et al. [18], could be explored to improve the DPAC-GAN architecture .

REFERENCES

[1] Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2016. An actor-critic algorithm for sequence prediction. *arXiv preprint arXiv:1607.07086* (2016).

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).

[3] 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*. 2672–2680.

[4] Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882* (2014).

[5] Vijay R Konda and John N Tsitsiklis. 2000. Actor-critic algorithms. In *Advances in neural information processing systems*. 1008–1014.

[6] Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2015. A diversity-promoting objective function for neural conversation models. *arXiv preprint arXiv:1510.03055* (2015).

[7] Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. 2016. Deep reinforcement learning for dialogue generation. *arXiv preprint arXiv:1606.01541* (2016).

[8] Jiwei Li, Will Monroe, Tianlin Shi, Sébastien Jean, Alan Ritter, and Dan Jurafsky. 2017. Adversarial learning for neural dialogue generation. *arXiv preprint arXiv:1701.06547* (2017).

[9] Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. 2017. DailyDialog: A Manually Labelled Multi-turn Dialogue Dataset. *arXiv preprint arXiv:1710.03957* (2017).

[10] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* (2015).

[11] Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out* (2004).

[12] 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. *arXiv preprint arXiv:1603.08023* (2016).

[13] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529.

[14] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, 311–318.

[15] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15, 1 (2014), 1929–1958.

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

[17] Richard S Sutton and Andrew G Barto. 1998. *Introduction to reinforcement learning*. Vol. 135. MIT press Cambridge.

[18] Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando de Freitas. 2016. Sample efficient actor-critic with experience replay. *arXiv preprint arXiv:1611.01224* (2016).

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

[20] Ronald J Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural computation* 1, 2 (1989), 270–280.

[21] Jingjing Xu, Xu Sun, Xuancheng Ren, Junyang Lin, Binzhen Wei, and Wei Li. 2018. DP-GAN: Diversity-Promoting Generative Adversarial Network for Generating Informative and Diversified Text. *arXiv preprint arXiv:1802.01345* (2018).

[22] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient.. In *AAAI*. 2852–2858.

[23] Weixiong Zhang. 1998. Complete anytime beam search. In *AAAI/IAAI*. 425–430.