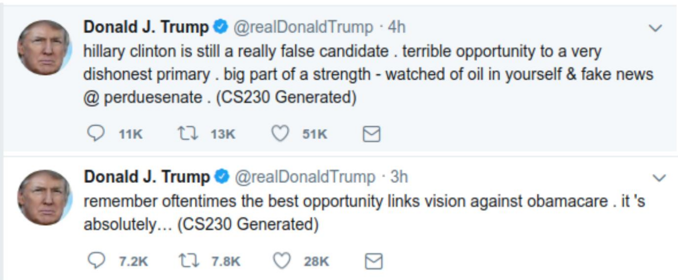
Generating Tweets using Generative Adversarial Networks

CS230

Ruowen Wang (SCPD)
ruowen19@stanford.edu




```
Layer (type)                 Output Shape              Param #
=================================================================
input (InputLayer)           (None, 15)                0
_________________________________________________________________
embedding (Embedding)        (None, 15, 100)           1000200
_________________________________________________________________
rnn_1 (CuDNNLSTM)            (None, 15, 128)           117760
_________________________________________________________________
rnn_2 (CuDNNLSTM)            (None, 15, 128)           132096
_________________________________________________________________
rnn_3 (CuDNNLSTM)            (None, 128)               132096
_________________________________________________________________
output (Dense)               (None, 10002)             1290258
=================================================================
Total params: 2,672,410
Trainable params: 2,672,410
Non-trainable params: 0
```

Figure 1: 3-layer LSTM baseline model

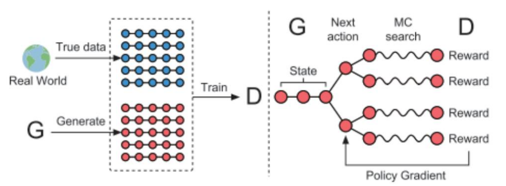


Figure 1: The illustration of SeqGAN. Left: D is trained over the real data and the generated data by G. Right: G is trained by policy gradient where the final reward signal is provided by D and is passed back to the intermediate action value via Monte Carlo search.

Abstract

In this work, we explore the possibility and challenges of applying Generative Adversarial Networks (GAN) to the task of generating tweets with a specific language style to simulate tweets from a real Twitter account. We use President Donald Trump's Tweets as a case study to train and evaluate whether a GAN-based text generation model can learn the language styles and generate realistic tweets. We also compare the GAN model with a basic RNN model as the baseline in terms of training difficulty and text generation quality. The preliminary results show that it is harder to train a GAN model than the baseline RNN model. The problems of training a GAN model, such as mode collapse, are observed. With fine-tuned hyperparameters, the generated texts from a GAN model are slightly better than the baseline RNN model. However, current results indicate that more efforts are needed on choosing better hyperparameters, finding better cost functions and designing better training strategy when applying GAN to text generation.

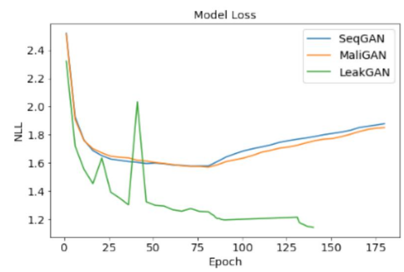


Table 1: Number of generated tweets in different qualities by three GAN models

Models	Make no sense	Grammar partially correct	Grammar mostly correct	Make sense semantically
SeqGAN	35	46	12	7
MaliGAN	44	32	16	8
LeakGAN	18	32	30	20