Environment

Linux, Intel i7, GTX 1060M(6G memory)

Package: tensorflow, nltk, genism, numpy

Model description & reward function

Embedding:pretrained Glove (https://nlp.stanford.edu/projects/glove/)

Seq2seq model:

Attention 加一般的 encoder-decoder model

RL model:

參考 Li, Jiwei, et al. "Adversarial learning for neural dialogue generation." 採用 GAN 的做法，其中 generator 同 seq2seq 使用的 model，而 discriminator 為 2 LSTM layers encoder + 1 LSTM layer decoder 並使用 attention

Reward function:

利用 Discriminator encoder 的輸出並經過一 projection layer 及 sigmoid 得到 reward，另外因 Paper 並無詳述其 baseline value 的做法，為使 reward 可為負值，簡單令 sigmoid 後的輸出 rescale 成 -1.0~1.0。

Performance

Model:

Seq2seq:baseline。

RL with GAN:

1. Teacher Forcing，以 GAN 訓練時亦會有一定程度的 seq2seq MLE 訓練生成器
2. 平均句子 tf-idf 降低常見句子權重

結果:失敗，無法重複得到 paper 的結果，到訓練後期判別器都能很好分辨句子，但生成器的句子全部會變得一模一樣，認為原因為下列兩點:

1. Paper 中的 baseline function 未清楚詳述其 real reward 為何，故無法實做，因而只單單使用 rescale 的方式而無法消除 reward 的 variance。

2. 因使用 Monte Carlo 來達到 Reward for Every Generation Step 需要非常久的訓練時間，所以直接使用 partially decoded 的 ground truth 及生成的句子來訓練判別器，而如 paper 中也說這樣的表現會較差。

Training data:

cornell corpus: nltk tokenize 句子後保留長度 5~30 的句子，在兩個 model 中回答的都很不相關。

opensub corpus: nltk tokenize 句子後保留長度 5~30 的句子，在 seq2seq model 回答稍微好一點。

call home+cornell corpus+opensub+twitter: 看不出來與 opensub 的差別，但最後使用這個設置。

Experiment settings and observation

Seq2seq model:

LSTM cell unit: 512

Train_Iters: 30000(~2 epochs) Batch_Size :64

Optimizer: Adam Loss: softmax loss Learning_Rate = 0.01

RL model:

Pre train **G** for 30000 iters using MLE

Pre train **D** for 30000 iters using training data and generated utterence

GAN:

D 與 G 的比例為 1:1，但訓練 G 時會有額外的 MLE 訓練，從 1:5 遞減到 1:0

但因到中期就訓練壞了所以事實上沒讓他整個跑完。

訓練時可以發現到即使有 MLE 的訓練，G 產生的句子還是會隨著訓練次數增加得到的 reward 越來越低，同時 D 的 loss 也越來越低，代表 D 能夠分辨真實跟生成的句子，不過 G 到最後儘管 condition 句子不同，G 只能產生相同的句子，因時間不足無法再做更多可能的改善。