Paper Reading

Wei Bi

Tencent AI Lab

2018 June 21

Adversarial Feature Matching for Text Generation

RNN-based autoencoder to generate a sentence:

- Fail to generate realistic sentences from arbitrary latent representations.
- Exposure bias by the use of RNN: error accumulates and quality decreases quickly after the first few words generate.

Adversarial Feature Matching for Text Generation

RNN-based autoencoder to generate a sentence:

- Fail to generate realistic sentences from arbitrary latent representations.
- Exposure bias by the use of RNN: error accumulates and quality decreases quickly after the first few words generate.

GAN to generate a sentence:

- Generator: map samples from a prior distribution to synthetic sentences that appears to be realistic.
- Discriminator: compare real and synthetic sentences.
- Two problems:
  - Model collapse: produce a single observation for multiple latent representation.
  - Vanishing gradient: as the discriminator gets better, the gradient of the generator vanishes.

Adversarial Feature Matching for Text Generation

RNN-based autoencoder to generate a sentence:

- Fail to generate realistic sentences from arbitrary latent representations.
- Exposure bias by the use of RNN: error accumulates and quality decreases quickly after the first few words generate.

GAN to generate a sentence:

- Generator: map samples from a prior distribution to synthetic sentences that appears to be realistic.
- Discriminator: compare real and synthetic sentences.
- Two problems:
  - Model collapse: produce a single observation for multiple latent representation.
  - Vanishing gradient: as the discriminator gets better, the gradient of the generator vanishes.

This paper considers a kernel-based moment-matching scheme to force the distribution of real and synthetic sentences to have matched moment in the latent-feature space.

TextGAN

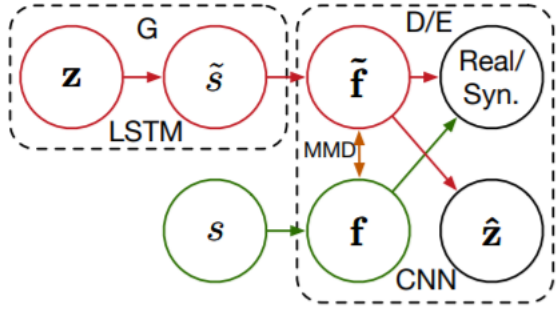

Figure 1. Model scheme of TextGAN. Latent codes z are fed through a generator $G(\cdot)$, to produce synthetic sentence $\tilde{s}$. Synthetic and real sentences ($\tilde{s}$ and s) are fed into a binary discriminator $D(\cdot)$, for real *vs.* fake (synthetic) prediction, and also for latent code reconstruction $\hat{z}$. $\tilde{\boldsymbol{f}}$ and $\boldsymbol{f}$ represent features of $\tilde{s}$ and s, respectively.

$$
\begin{aligned}
L_{GAN} &= E_{s\sim S} \log D(s) + E_{z\sim p_z} \log[1 - D(G(z))] \\
L_D &= L_{GAN} - \lambda L_{recon} + \lambda L_{MMD^2} \\
L_G &= L_{MMD^2} \\
L_{recon} &= \|\hat{z} - z\|^2 \\
L_{MMD^2} &= \|E_{f \text{ from real}} \phi(f) - E_{\hat{f} \text{ from fake}} \phi(\hat{f})\|^2_{\mathcal{H}} \\
&\quad \text{(mean squared difference between two sets of samples))}
\end{aligned}
$$

MMD

MMD measures the mean squared difference between two sets of samples X and Y over a Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}$, $\phi(x) \in \mathcal{H}$ is the feature mapping, $k(x, y)$ is the kernel function.

$$\begin{aligned} L_{MMD^2} &= \|E_{x\sim X}\phi(x) - E_{y\sim Y}\phi(y)\|^2_{\mathcal{H}} \\ &= E_{x\sim X}E_{x'\sim X}[k(x, x')] + E_{y\sim Y}E_{y'\sim Y}[k(y, y')] \\ &\quad -2E_{x\sim X}E_{y\sim Y}[k(x, y)] \end{aligned}$$

- With a universal kernel like the Gaussian kernel $k(x, y) = \exp(-\frac{\|x-y\|^2}{2\sigma})$ with bandwidth σ, minimizing the MMD objective will match moments of all orders.
- MMD forces the generator to produce highly diverse sentences to match the variation of real sentences, by latent moment matching, thus alleviating the mode-collapsing problem.
- MMD is a proper metric when the kernel is universal. If the kernel function is universal, the MMD metric will be no worse than the Total Variance Distance in terms of vanishing gradients.

Experiments

- The proposed model is trained using a combination of two datasets to investigate whether it can generate sentences that integrate both scientific and informal writing styles.

Table 3. Intermediate sentences produced from linear transition between two points (A and B) in the latent feature space. Each sentence is generated from a latent point on a linear path.

	textGAN	**AE**
A	our methods apply novel approaches to solve modeling tasks .	
-	our methods apply novel approaches to solve modeling .	our methods apply to train UNK models involving complex .
-	our methods apply two different approaches to solve computing .	our methods solve use to train) .
-	our methods achieves some different approaches to solve computing .	our approach show UNK to models exist .
-	our methods achieves the best expert structure detection .	that supervised algorithms show to UNK speed .
-	the methods have been different related tasks .	that address algorithms to handle) .
-	the guy is the minimum of UNK .	that address versions to be used in .
-	the guy is n't easy tonight .	i believe the means of this attempt to cope .
-	i believe the guy is n't smart okay?	i believe it 's we be used to get .
-	i believe the guy is n't smart .	i believe it i 'm a way to belong .
B	i believe i 'm going to get out .	

Conclusions

- This model delivers superior performance compared to related approaches, can produce realistic sentences.
- The learned latent representation space can “smoothly” encode plausible sentences.
- The response generation model(Seq2seq) also suffers from the mode collapse problem, i.e. multiple input representations will output the same generic response sequence. If GAN is not used, can we still apply the MMD idea to constrain the input representation space for a better representation space?

Sequence Tutor: Conservative Fine-Tuning of Sequence Generation Models with KL-control

This paper proposes a general method for improving the structure and quality of sequences generated by Seq2seq.
To apply RL to sequence generation:

- Generating the next token in the sequence is treated as an action a.
- The state of the environment consists of all of the tokens generated so far, i.e. $s_t = \{a_1, a_2, \ldots, a_{t-1}\}$
- Given action a_t, we would like the reward r_t to combine information about the prior policy $p(a_t|s_t)$ as output by the Reward RNN, as well as some domain- or task-specific rewards r_T.

DQN

Given the state of the environment at time t, s_t, the agent takes an action at according to its policy $\pi(a_t|s_t)$, receives a reward $r(s_t, a_t)$, and the environment transitions to state, s_{t+1}. The optimal deterministic policy π^* satisfies the Bellman optimality equation

$$Q(s_t, a_t, \pi^*) = r(s_t, a_t) + \gamma E_{p(s_{t+1}|s_t,a_t)}[max_{a_{t+1}} Q(s_{t+1}, a_{t+1}; \pi^*)]$$

DQN approximates $Q(s, q; \theta)$ by a DNN:

$$L(\theta) = E_\beta[(r(s, a) + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta))^2]$$

- β is the exploration policy.
- θ^- is the parameters of the target Q-network that is held fixed during the gradient computation.

Sequence Tutor

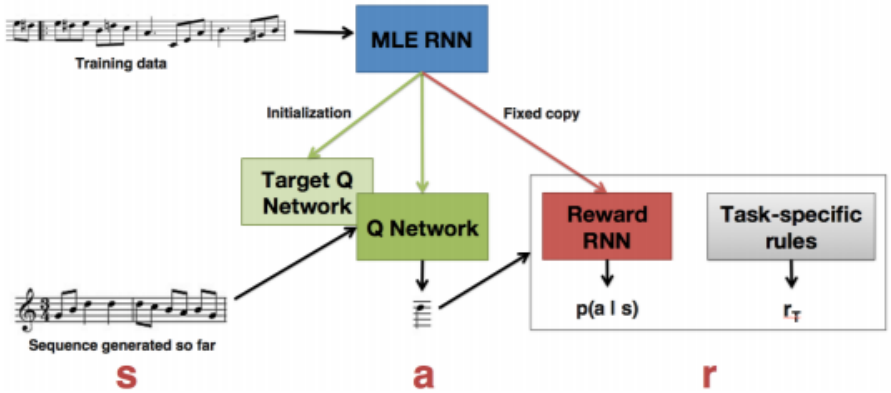

Figure 1: An RNN pre-trained on data using MLE supplies the initial weights for the Q-network and target Q-network, and a fixed copy is used as the Reward RNN.

- Pretrain a Seq2seq and fix it as a Reward RNN.
- Copy the pretrained Seq2seq network as the Target Q Network and Q network for the DQN learning.
- The reward at time t: $r(s,a) = \log p(a|s) + r_T(a,s)/c$.
- The ojbective and learned policy of DQN:

$$L(\theta) = E_\beta[\log p(a|s) + r_T(a,s)/c + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s,a;\theta))^2]$$

$$\pi_\theta(a|s) = \delta(a = \arg\max Q(s,a;\theta))$$

Sequence Tutor...

- DQN learns a deterministic policy, not be ideal for sequence generation.
- The problem can be expressed as a KL control problem for a non-Markovian system.
- They treat a trained MLE sequence model as the prior policy, and thus the objective is to train a new policy to maximize some rewards while keeping close to the original MLE model.
  - $\tau = \{a_1, a_2, \ldots, a_{t-1}\}$: the sequence, $\gamma(\tau)$: the reward of the sequence, $p(\tau)$: the prior distribution over τ given by the trained sequence model, $q(\tau)$: the policy of the Sequence Tutor model:

$$L(q) = E_{q(\tau)}[\gamma(\tau)/c - D_{KL}[q(\tau)||p(\tau)].$$

  - The reinforcement learning objective

$$L(\theta) = E_{\pi}[\sum_{t} r(s_t, a_t)/c + \log p(a_t|s_t) - \log_{\pi_\theta}(a_t|s_t)]$$

    $E_{\pi}[\cdot]$:expectation with respect to sequences sampled from π.
  - Derive two algorithm to parameterize π_θ.

Experiments

- Generation of Melody and Molecular
- Compare three methods for implement the Sequence tutor:
  - Q-learning with the deterministic policy.
  - two methods for KL-control with the non-deterministic policy.
- Compare the RL-only with no prior policy and MLE RNN.

Conclusions

- Similar methods can be applied on text generation if a deterministic policy is applied.