

Controlled Generation of Text

Agustinus Kristiadi and Yonathan Santosa

University of Bonn

1 Feb 2017

Outline

Motivation

Background

- Background on Generative Models

- GAN

- VAE

Papers Review

- Generating Sentences from a Continuous Space

- Toward Controlled Generation of Text

Implementation

Results

Conclusion

Reference

Motivation from Computer Vision

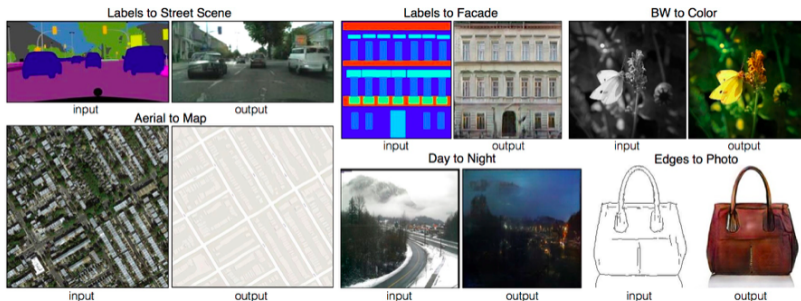


Figure 1: Pix2Pix. Isola, et. al., 2016

Generative Modeling in NLP

Can we also do generative modeling in NLP?

Varying the unstructured code z

(*“negative”, “past”*)

the acting was also kind of hit or miss .

i wish i 'd never seen it

by the end i was so lost i just did n't care anymore

(*“negative”, “present”*)

the movie is very close to the show in plot and characters

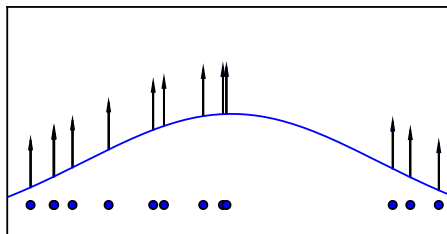
the era seems impossibly distant

i think by the end of the film , it has confused itself

Table 1: From Hu, 2017

Generative Models

- ▶ Density estimation problem: given data samples (empirical distribution), find the true distribution generating those samples.
- ▶ Based maximum likelihood principle.
- ▶ But, likelihood function is often intractable! \implies This problem is hard!



$$\theta^* = \arg \max_{\theta} \mathbb{E}_{x \sim p_{\text{data}}} \log p_{\text{model}}(x | \theta)$$

Figure 2: From Goodfellow, 2016.

Generative Adversarial Networks: GAN

- ▶ Two players game, both are neural networks:
 - ▶ G : generate data samples.
 - ▶ D : inspect data samples.
- ▶ Objective: G fools D ; D resists G .
- ▶ Thus, this is minimax game \implies the solution is Nash equilibrium.
- ▶ GAN will not work on discrete data as it is nondifferentiable.

Variational Autoencoder: VAE

- ▶ Idea: instead of directly works with likelihood function, works with relaxed lower bound instead.
- ▶ This lower bound comes from variational mean field theorem, so we call it variational lower bound:

$$\log P(X) \geq E[\log P(X|z)] - D_{KL}[Q(z|X)||P(z)]$$

- ▶ This lower bound is easier to work with as it decomposes into two easy terms:
 - ▶ First is reconstruction term.
 - ▶ Second is regularization term for z .
- ▶ We use neural networks for $Q(z|X)$ and $P(X|z)$ thus essentially this is an autoencoder.
- ▶ VAE works for discrete data out of the box.

Generating Sentences from a Continuous Space, Bowman 2016

- ▶ Based on VAE, with both encoder and decoder to be LSTMs.
- ▶ Very straight-forward.
- ▶ Latent variable z is seen as the global conditioning code for the decoder. It captures the semantic of text.

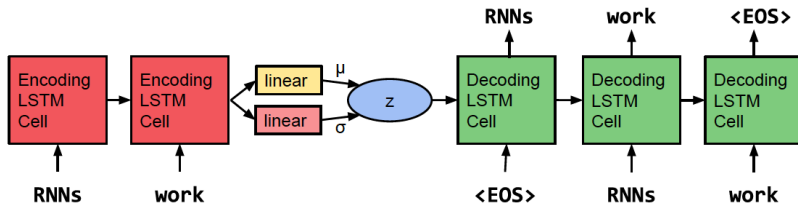


Figure 3: Model of Bowman, et. al., 2016

Generating Sentences from a Continuous Space, Bowman 2016

- ▶ Difference with normal autoencoder: z is stochastic, i.e. represented by some posterior distribution $Q(z|X)$, this we regularize to be close to prior $N(0, I)$.
- ▶ $P(X|z)$ is a “continuous” function, i.e. changing z a little bit implies X also changed only a little bit.

Toward Controlled Generation of Text, Hu 2017

- ▶ Combining VAE+GAN.
 - ▶ Base model is Bowman, 2016.
 - ▶ Add a discriminator on top to learn conditioning code c .
 - ▶ Use softmax trick to alleviate non-differentiability.
- ▶ Conditioning code could be anything, e.g. sentiment, tense labels.
- ▶ Thus text is generated based on its semantic z and its label c .

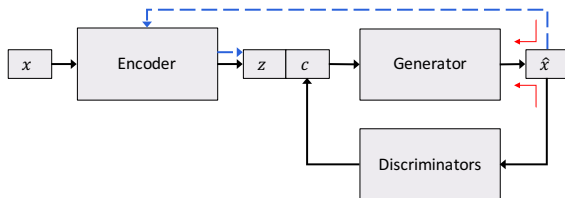


Figure 4: From Hu, 2017

CGT: Controlled Generation of Text

- ▶ Based on Hu, 2017 and Bowman, 2016.
- ▶ Consists of three components: Encoder, Decoder, and Discriminator.
- ▶ We pre-train Encoder and Decoder first as in Bowman, 2016.
- ▶ Then we train all of them as in Hu, 2017.

CGT: Discriminator

- ▶ Following architecture by Kim, 2014.

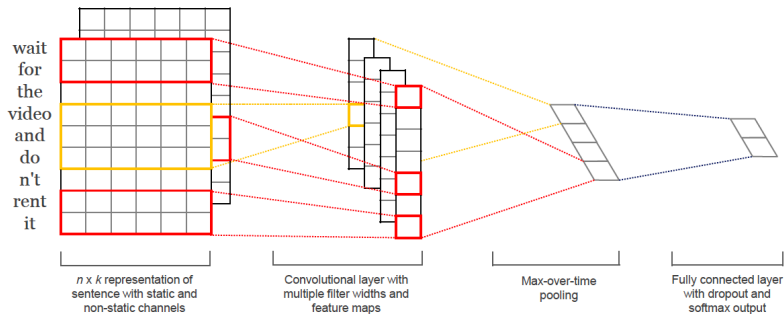


Figure 5: From Kim, 2014

CGT: Training Base VAE

- ▶ Training similar to normal VAE, but with two tricks to help stabilizing it.
- ▶ First is KL weight annealing, to make sure KL term in VAE loss to be above zero.
- ▶ Second is word dropout, i.e. with prob. p , set word in decoder input to $\langle UNK \rangle$

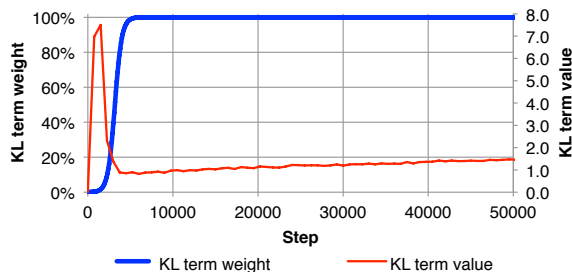


Figure 6: From Bowman, 2016

CGT: Losses

- ▶ Encoder: Same as VAE loss.
- ▶ Decoder:

$$L_{Attr,z}(\theta_G) = \mathbb{E}_{p(z)p(c)} \left[\log q_E(z | \tilde{G}_\tau(z, c)) \right]$$

$$L_{Attr,c}(\theta_G) = \mathbb{E}_{p(z)p(c)} \left[\log q_D(c | \tilde{G}_\tau(z, c)) \right]$$

$$L_G = L_{VAE} + \lambda_c L_{Attr,c} + \lambda_z L_{Attr,z}$$

- ▶ Where $\tilde{G}_\tau(z, c)$ is a soft embedding, i.e. taking expected embedding instead of doing argmax.

CGT: Losses

- Discriminator:

$$L_s(\theta_D) = \mathbb{E}_{x_L} [\log q_D(c_L|x_L)]$$

$$L_u(\theta_D) = \mathbb{E}_{p_G(\hat{x}|x,c)p(z)p(c)} [\log q_D(c|\hat{x}) + \beta \mathcal{H}(q_D(c'|\hat{x}))]$$

$$L_D = L_s + \lambda_u L_u$$

- Where \mathcal{H} is empirical Shannon Entropy and β , λ_u are weighting hyperparameters.

CGT: Pseudocode

► Training overview:

1. Input = Corpus with labels $\{(x_L, c_L)\}$
2. Initialize base VAE by minimizing L_{VAE} on x_L with c sampled from prior $p(c)$. (Bowman, 2016)
3. Repeat:
 - 3.1 Train encoder by minimizing L_{VAE}
 - 3.2 Train decoder by minimizing L_G
 - 3.3 Train discriminator by minimizing L_D
4. Until convergence

Experiment Setup

- ▶ Dataset: SST dataset, filtered out neutral sentiment and any sentence with length > 15 . Total sentences are 2837.
- ▶ Library: PyTorch.
- ▶ Training procedure:
 - ▶ Train base VAE as in Bowman, 2016
 - ▶ Train discriminator as in Hu, 2017
- ▶ Full code is available at: <https://github.com/wiseodd/controlled-text-generation>.

Latent Interpolation

- ▶ Given z_1 and z_2 and $0 \leq \alpha \leq 1$; $z_\alpha = \alpha z_1 + (1 - \alpha)z_2$.
- ▶ Given z_α , we can generate sentence: $\hat{x} = \text{Generator}(z_\alpha)$
- ▶ We do this for several α from 0 to 1.

the film is the courage of the unsalvageability of the year .

the film is full of the year 's best .

the film 's unhurried pace is a lot of nada .

the film 's unhurried pace is repulsive and unfocused .

the film 's unhurried pace is actually one of anything .

Table 2: Interpolation result of our model.

Conditional Generation

- ▶ Given a fixed $c = \{\text{"positive"}, \text{"negative"}\}$, generate sentences by varying z .

Varying the latent variable z

(*"positive"*)

the film is full of the year 's best .

a deftly entertaining thriller .

a lovably movie .

(*"negative"*)

the actors are no chemistry or unappealing to care one .

delivers a film living pretty bad movie .

full of flatulence jokes and clichéd to cliches .

Table 3: Result of conditional generation

Failure Cases

- ▶ We trained the model on very small dataset (around 2800 sentences). Clearly this is not enough to combat overfitting.
- ▶ Examples of failure cases:
 - ▶ **positive:** “includes all the end all the wrong reasons besides .”
 - ▶ **negative:** “a processed comedy chop suey .”

Conclusion

- ▶ VAE is straight-forward for text, GAN not so much.
- ▶ Needs tricks to make the approach works.
- ▶ Conditional Generation of Text model combines GAN and VAE.
- ▶ With a trained model, we can generate new sentence with desireable property.
- ▶ Experiment with larger dataset should give better results.

Reference

- ▶ Bowman, Samuel R., et al. "Generating Sentences from a Continuous Space." CoNLL 2016. [pdf]
- ▶ Hu, Zhiting, et al. "Toward controlled generation of text." ICML 2017. [pdf]
- ▶ Goodfellow, Ian. "NIPS 2016 tutorial: Generative adversarial networks." arXiv preprint arXiv:1701.00160 (2016).