Controllable text generation

Boris Kovalenko
SUID 06201315
kboris@stanford.edu

1. Introduction

Since the introduction of deep generative models a few years ago, there was a great advancement in this area, especially in the visual domain. On the contrary, tasks in natural language generating have been less studied. Producing realistic sentences is a challenging task for generative models as they required to capture complex semantic structures within given training text corpus. For controllable text generation, there are additional challenges. First is - text samples are discrete and as a result non-differentiable. This doesnt allow the use of global discriminator, which is common in the generative models in the visual domain. For example in generative adversarial networks used for image generation. Another challenge for controllable text generation relates to learning disentangled latent representations. Parts of a latent vector, obtained from encoded text sample or sampled from latent space, are not mutually independent. Additionally, they are hard to relate to semantic properties in generated text samples. Varying individual parts of latent vector representation can cause unpredictable results in the generated text sample.

Area of generic and controllable text generation using deep neural models is not very well explored. Recently few approaches were introduced. Researchers use variational autoencoders [1] and generative adversarial networks [8] to solve this task and results were not very good. Recently new advancements in this area were made, with impressive results [3][7]. The [3] proposes a model that addresses issues stated above. The new approach is still based on generative variational autoencoder but uses additional building blocks and advanced training schemes, which allow the model to achieve plausible results with a benefit of semantic property control.

In this project, I'm working on implementing models described in [3] with focus on conditional text generation. Specifically, I'm interested in replicating model capable of text generation given a sentiment label. Approach for plausible text generation with predetermined sentiment can have lots of important applications. For example enriching chatbots answers with the sentiment, tense and other semantic properties can drastically improve the experience of interaction with them. Internet companies which rely on human-generated reviews in their business can enhance interaction of people with their knowledge base. And many others.

2. Related work

Variational autoencoders (VAEs) [4] were introduced in 2013. The model consists of encoder and decoder neural networks. The encoder encodes a data sample to a latent representation. The decoder is capable of data sample generation using latent representation, which was prepared by the encoder or sampled from latent space. VAE's are trained by maximizing a variational lower bound on the data log-likelihood under the generative model. To match the posterior of the latent code with a prior a KL divergence loss is minimized. This makes possible to produce data sample from any latent code sampled from the prior. Without the KL minimization, VAEs becomes simple autoencoders, which is not a generative model.

To train VAE for controllable text generation - the wake-sleep algorithm is used [2]. This algorithm was introduced for training deep directed graphical models. The wake phase updates the decoder with samples generated from the encoder network using training data. In the sleep phase, the encoder network is updated using samples produced by the decoder.

Generative models use discriminator model feedback for generated samples assessment. Provided feedback is used to update the generative model and push it into producing samples which are "preferred" by the discriminator. Applying discriminators to text generation directly is not possible due to the non-differentiability of discrete samples. All introduced deep text generative models [1][8][5][6] are not using discriminators and not impose disentangled property for latent representation. As a result generation of text samples with predefined semantic property is not possible.

The model described in [6] is capable of text generation with sentiment control. The model is multiplicative LSTM (mLSTM) trained for the task of next character prediction. After mLSTM is trained, a linear combination of learned recurrent units is taken to produce single sentiment classifier (sentiment neuron), capable of text generation. The model

produces plausible text samples and sentiment is controllable. Drawbacks of this approach are a demand to overwrite sentiment neuron, obtained from mLSTM, to change sentiment polarity. And lack of ability to produce text samples with multiple controllable sentiment properties.

3. Dataset

For training VAE we use IMDB Movie Review Dataset. The dataset comprises 50000 reviews for popular movies. Each review is labeled either as positive or as negative. Dataset is balanced and has 25000 positive and negative reviews.

To train model we use 10000 most popular words, rest of vocabulary is replaced with token $< unk >$. To ease text generation modeling we impose a limit on the length of sentences. The maximum length of sentences is set to 15. Sentences longer than 15 words are truncated, those which have less than 15 words are padded with token $< pad >$. Additionally, to give the model sense of beginning and end, each sentence is padded at the beginning and at the end with tokens $< start >$ and $< end >$.

For neural network models, some representation of text is needed. A simple bag of words model is not suitable for this kind of models. Usually, word vectors are used. Models like Word2Vec or Glove produce a good dense representation of words, which captures linguistic properties and semantic concepts from text corpus it was trained on. Alternatively, word vectors could be learned jointly with main model training. Usually, use of pretrained word vectors provides speed up in training and possibly improves main model quality. But if there is enough data for training this difference could be negligible. To train VAE we chose the second approach and learn word vectors jointly with main model training.

Below are examples of movie reviews from the used dataset, there is 1 positive and 1 negative movie review:

- Liked Stanley & Iris very much. Acting was very good. Story had a unique and interesting arrangement. The absence of violence and sex was refreshing. Characters were very convincing and felt like you could understand their feelings. Very enjoyable movie.

- Not only is it a disgustingly made low-budget badacted movie, but the plot itself is just STUPID!!! A mystic man that eats women? (And by the looks, not virgin ones) Ridiculous!!! If you've got nothing better to do (like sleeping) you should watch this. Yeah right.

4. Methods

The model described in [3] is generative model, used for text generation with predefined sentiment property. It's a variational autoencoder trained with the extended wake-sleep procedure. Model description is below, the model scheme is in figure 1.

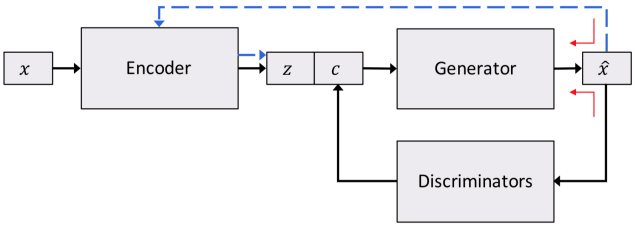

Figure 1. VAE used for controllable text generation. Given observation x *Encoder* infers latent vector z. Which is a representation of x in latent space. Given latent code (z, c) *Generator (Decoder)* produces plausible text sample, which possesses semantic property, set by c. Discriminator returns probability of semantic attribute in given text sample.

As any autoencoder, the variational autoencoder has two parts encoder and decoder. The encoder is LSTM-RNN followed by dense layers. Encoder embeds sentences into two dense vectors μ and σ^2. Both vector calculated using final hidden state of encoder's LSTM. Using this two vectors we can calculate vector z, through reparametrization trick:

$$z = \mu(x) + \sigma^2(x) * \epsilon$$

where $\epsilon \sim N(0, 1)$. To control sentence sentiment (or other semantic property), we allocate one dimension of the latent representation to encode sentiment. Such design makes possible to learn disentangled latent representation. And generated samples sentiment could be controlled simply by specifying a particular code.

Second part of model is decoder or generator, since it can be used without encoder. Generator model is LSTM-RNN as well. Usually in text generation, generator learns to reconstructs sequence $\hat{x}_i = \{\hat{x}_1, \hat{x}_2, ...\hat{x}_T\}$ with conditioning on z_i. Since we want controllability for produced sequence we introduced structured part to z_i, vector c, which is vector of one-hot labels for different properties which are desired for generated text. For example if we want sentiment to be controllable and we have two classes: positive and negative, we could use vector c of size two with positive class depicted as $c = [0, 1]$ and negative class $c = [1, 0]$. To generate text sample, generator conditions on both vectors z_i and c_i:

$$\hat{x} \sim G(z, c) = p_G(x|z, c) = \prod_t p(\hat{x}_t|\hat{x}_{<t}, x, c)$$

Because generator is LSTM, to produce element $\hat{x}_t$ of sequence at time t, we also condition on past by using "old" elements of generated sequence $\hat{x}_1, \hat{x}_2, ...\hat{x}_{t-1}$.

The third element of the model is the discriminator. The discriminator is a classifier, which outputs class probabilities:

$$D(x) = q_D(c|x)$$

We have one discriminator for each semantic property. For example, if we introduce control of sentiment, our discriminator is a sentiment classifier. The discriminator is a convolutional neural network trained as a binary classifier.

Model is trained using the extended wake-sleep procedure. Discriminator, encoder, and generator are trained in step by step manner. The training algorithm description is below.

Algorithm 1 VAE training approach

Input: A large corpus of unlabeled sentences $X = \{x\}$
A few sentence attribute labels $X_L = \{(x_L, c_L)\}$
Parameters: $\lambda_c, \lambda_z, \lambda_u, \beta$ balancing parameters
VAE warming up: Initialize VAE by minimizing reconstruction and KL Loss on X with c sampled from prior $p(c)$
Repeat until convergence
[1] Train the discriminator D
[2] Train the generator G
[3] Train encoder E
Output: VAE, which has sentence generator G capable of generating text samples with condition on (z, c)

In the first step, VAE initialization during training reconstruction loss and KL loss is minimized. KL loss part is needed to force latent encoder to be close to the prior $p(z)$, where $p(z) \sim N(0, 1)$. Below is loss minimized during VAE initialization:

$$L_{VAE} = -KL(q_E(z|x)||p(z)) + E_{q_E(z|x)q_D(c|x)}[\log p_G(x|z,c)]$$

After VAE is trained for few epochs, next step is a loop within which discriminator, generator, and encoder are trained. First updates for the discriminator. The discriminator is a classifier, which is trained on the labeled dataset, as training objective cross-entropy loss is used.

$$L_D = E_{x_L}[\log q_D(c_L|x_L)]$$

To provide additional learning signal for generator, which enforces it to produce text samples with semantic attribute c, following loss term is used in generator loss:

$$L_{Attr,c} = E_{p(z),p(c)}[\log q_D(c|\hat{G}_\tau(z,c))]$$

To force disentangled representation we add following loss term is used in generator loss:

$$L_{Attr,z} = E_{p(z),p(c)}[\log q_E(z|\hat{G}_\tau(z,c))]$$

where encoder used to produce latent code z from generated text sample $\hat{G}_\tau(z, c)$

Combining all loss terms, we get generator loss:

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

Finally, we make the additional step to update encoder and decoder, by minimizing loss L_{VAE} one more time. This training procedure continues, until convergence. This training procedure is depicted in figure 2.

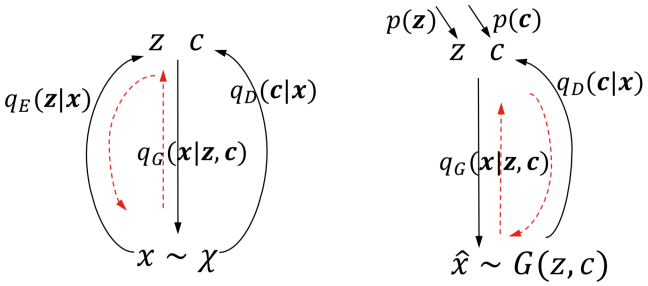

Figure 2. On the left VAE initialization and encoder update step depicted. On the right discriminator and generator models are updated.

5. Results

The model was trained in 2 stages. First VAE was warmed up for 10 epochs, after that training continued for 90 more epochs, using the wake-sleep procedure. Below are plots for loss values for VAE and discriminator for the second stage.

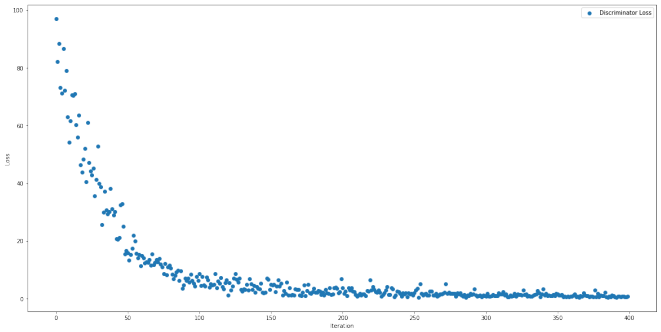

Figure 3. Discriminator training loss

As we see discriminator model achieves very low values of loss on the training set.

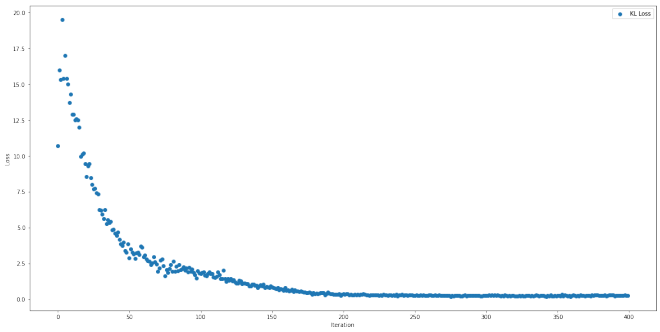

Figure 4. KL loss

KL loss also looks good. That means our encoder is well trained to produce latent vectors that are very close to samples from $N(0, 1)$

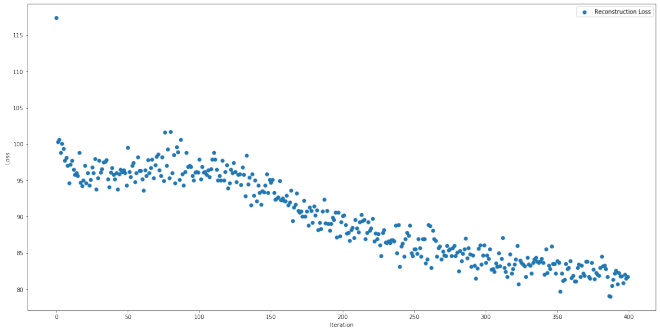

Figure 5. Reconstruction loss of VAE

Reconstruction loss of VAE goes down as training progresses. We can expect to see plausible examples of generated text samples. Examples of generated text samples are below.

Good examples generated using latent vector sampled from prior distribution:

- *Positive* - i don't know what i can say about this movie is a charming movie
- *Positive* - one of the most respected films i have seen in tears of the <unk>
- *Negative* - what can i say that this is the worst movie i have ever seen
- *Negative* - if you have to say this is one of the worst movies i have
- *Neutral* when i first saw it on dvd for the first time i saw it
- *Neutral* i have never heard of this movie i have to say that it was

Failure cases, examples generated using samples from prior distribution:

- the <unk> of the <unk> of the <unk> <unk> of the <unk> of the <unk>
- the makers of the makers of the makers of the makers of this movie is

Good examples generated using the latent vector, produced by encoder from input:

- *Input* - the movie is not bad acting is bad
- *Positive output* - this is one of the best movies i have ever seen it was a
- *Negative output* - this is one of the worst movies i have ever seen it makes me
- *Input* - his acting was very good
- *Positive output* - one of the most respected movies i've seen in tears for the first time
- *Negative output* - this is a stupid movie i have ever seen it was the first time i

Failure cases, examples generated using the latent vector, produced by encoder from input:

- *Input* - i like movie plot it is very original
- *Positive output* - this is one of the most movies i have seen in a <unk> and it
- *Negative output* - if you are looking for a bunch of movies that you have to be
- *Input* - i consider movies based on comics interesting
- *Positive output* - i am a huge fan of <unk> <unk> movies and this is one of the
- *Negative output* - i have never heard of this movie i am a huge fan of the <unk>

As we can see from examples generated using prior distribution and encoded inputs, good results look plausible. Failure cases mostly relate to the model inability to infer correct sentiment for encoded input or generic text generation failure. Both problems could be caused by overfitting training dataset or by problems in model optimization. Since our model has discriminator and VAE tries to "persuade" it by generating examples discriminator likes, optimization process becomes unstable and requires careful parameter selection.

6. Future work

In project controllable text generation VAE model with discriminator is implemented. Implemented model is capable of generation of plausible text samples with the predefined sentiment. It would be beneficial to push the idea of controllable text generation further. Generating text with respect to context. Context can be represented by a piece of text and labels (for example given news story headline and sentiment label model generates comments for the story). Such model will require the presence of a mechanism for context encoding.

A very common problem for deep generative models is mode collapse. Mode collapse is a problem of decrease of size for support of latent space, as it becomes small all generated examples become very similar if not the same. It is a challenge to identify mode collapse in the visual domain. On the contrary, since it is easier to measure similarity for text, studying mode collapse for text generating model is easier but equally important, since it could be possible to introduce an important metric for model quality.

References

[1] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Józefowicz, and S. Bengio. Generating sentences from a continuous space. *CoRR*, abs/1511.06349, 2015.

[2] G. E. Hinton, P. Dayan, B. J. Frey, and R. M. Neal. The "wake-sleep" algorithm for unsupervised neural networks. *Science*, 268 5214:1158–61, 1995.

[3] Z. Hu, Z. Yang, X. Liang, R. Salakhutdinov, and E. P. Xing. Toward controlled generation of text. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1587–1596, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

[4] D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. *ArXiv e-prints*, Dec. 2013.

[5] M. J. Kusner and J. M. Hernández-Lobato. GANS for Sequences of Discrete Elements with the Gumbel-softmax Distribution. *ArXiv e-prints*, Nov. 2016.

[6] A. Radford, R. Józefowicz, and I. Sutskever. Learning to generate reviews and discovering sentiment. *CoRR*, abs/1704.01444, 2017.

[7] Z. Yang, Z. Hu, R. Salakhutdinov, and T. Berg-Kirkpatrick. Improved variational autoencoders for text modeling using dilated convolutions. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3881–3890, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

[8] L. Yu, W. Zhang, J. Wang, and Y. Yu. Seqgan: Sequence generative adversarial nets with policy gradient. *CoRR*, abs/1609.05473, 2016.