COMS6998 Practical Deep Learning Systems Performance Final Project Report

WangYao

WY2353

Motivation and Background

Just like works of Old English, works of Old Chinese are always more exquisite and shorter than Modern Chinese. One of the most beautiful Chinese literature styles is ancient Chinese poetry, which can be compared to Sonnet from Shakespeare in Old English. It always has strict stylistic structure and a word can be given a deep meaning. Among all these, the two most strict and common styles are five-word-poem and seven-word-poem. Five-word poem is basically

’blah blah blah blah blah, blah blah blah blah blah.’
’blah blah blah blah blah, blah blah blah blah blah.’

Where as seven-word-poem always looks like:

’blah blah blah blah blah blah blah, blah blah blah blah blah blah blah.’
’blah blah blah blah blah blah blah, blah blah blah blah blah blah blah.’

Which means each sentence of five is made of a comma in the middle of ten words and a full stop in the end. While the seven-word poem means each line is made of a comma in the middle of fourteen words and a full stop in the end. Also, a five word or a seven word poem is always made by sentences of a even number. Mostly, there are two sentences in a five word poem and two/four sentences in a seven word poem.   For better understanding, I will give an example and translate it into English

舉頭望明月，低頭思故鄉

春風又綠江南岸， 明月何時照我還。

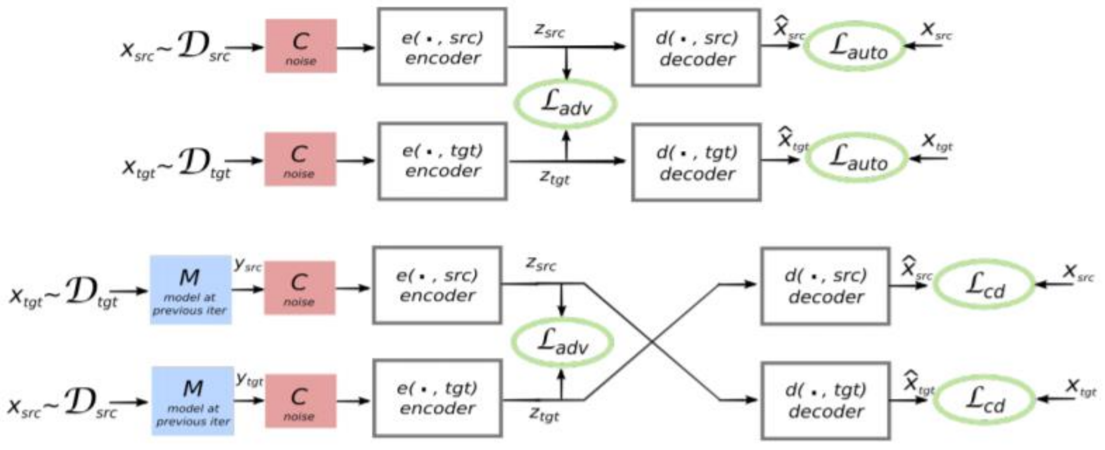
Above are two poems that should be the very first poem we learned when we are about 5 or 6 years old. Here the first line is a five-word poem, which means ’Look up at the moon, look down at my home.’ And the second line is a sentence from seven-word poem meaning ’Another year spring green the river bank, when will the moon bring me back’. For your better understanding Here the translation is direct and the has the same number of words that can parallel with the of original Chinese poem.

Here both poems show the author's homesickness and both poets show their emotion by the description of the moon, since ancient Chinese believe that every single part of the China shares the same moon at the same time and the moon will lead them home no matter where they are. To some extent, we can say that each of the sentence above is the synonymous rewriting of the other one and this kind of text style transfer is what we want. Imagine if we could rewrite a cultured and refined sentence made by our ancestor into another beautiful sentence, think about it, that would be amazing.

Goal

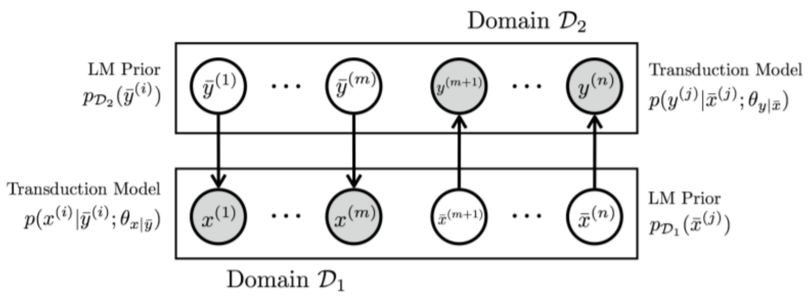
My goal is to build a transduction model that can rewrite a sentence from five-word poem into a sentence seven-word poem. Consider different numbers of sentences in a poem, rewrite poem to a poem seems difficult. So here I choose the transduction from a sentence to another sentence. If the output is just like the example above, I'd say I got a successful model.
Now I believe that this goal can be considered in 3 parts, meaning we should evaluate the performance in 3 way. First, let's say we're converting a five to a seven, our model should correctly output a seven-word poem, which means the comma must be at the correct position (8th) and the stop should be at the 16th blank of our generative sentence. Second, the generative sentence should be an understandable sentence, meaning it should be similar to human language. We can evaluate this by computing the perplexity on test data. Third, the generative sentence should be synonymous with our input sentence. However, this is hard to evaluate and I cannot come out with some better ideas rather than evaluate by myself.

Background work introduction

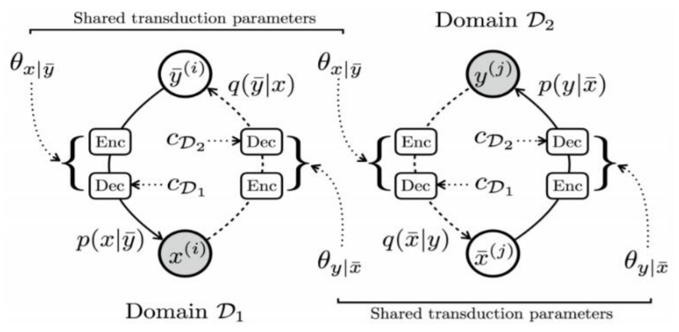


Model structure in *Unsupervised machine translation using monolingual corpora only*

We notice that my poem style transfer work is similar to unsupervised machine translation. Since we don't have much labeled paralleled corpus, unsupervised machine translation is an important topic. One of paper from this area is the *Unsupervised machine translation using monolingual corpora only* by Facebook in 2018. The paper used implement a discriminator just like GAN. The paper assumes that there exists a monolingual corpus on each language and wants to parallel the unpaired corpus in hidden states. The structure of this model basically is an encoder decoder structure, where we call it a denoising autoencoder. In this paper the encoder and decoder LSTM and attention part share the same parameters but different parameters in two word embeddings. Also a discriminator is used to identify the hidden state is which domain. The training then can be considered into 3 parts, first the denoising autoencoding training, which means you need to reconstruct your input sentence. Second the cross-domain training, which means you need to reconstruct X from a $\hat{X} = \mathrm{f}(y)$, where the $\widehat{\mathrm{X}}$ is the translation result from your model. Third the adversarial training, which means you need to train your discriminator.



Another important work is *A Probabilistic Formulation of Unsupervised Text Style Transfer* by Junxian He in 2020. The model proposed a method that treats an observed non-parallel text corpus as a partially observed parallel corpus. $\mathrm{X} = \mathrm{x}^{(1)}, \mathrm{x}^{(2)}, \cdots, \mathrm{x}^{(m)}$ represent observed data from domain D1, while we let $Y = \mathrm{y}^{(m+1)}, \mathrm{y}^{(m+2)}, \cdots, \mathrm{y}^{(\mathrm{n})}$ represent observed data from domain D2. Thus, none of the observed sentences share indices. the model, introduce latent sentences to complete the parallel corpus. Specifically, ˆ$\{X\} = \{\hat{x}^{(m+1)}, \hat{x}^{(m+2)}, \cdots, \hat{x}^{(n)}$ represents the set of latent parallel sentences in D1, while , $\widehat{\mathrm{Y}} = \hat{y}^{(1)}, \hat{y}^{(2)}, \cdots, \hat{y}^{m}$represents the set of latent parallel sentences in D2. Then the goal of unsupervised text transduction is to infer these latent variables conditioned the observed non-parallel corpora; that is, to learn $p(\hat{y}|x)$ and $p(\hat{x}|y)$.



The whole structure: First you need to train two language models for Domain1 and Domain2 respectively. The language model is used for compute the probability of an input sentence in this domain. And we will talk about this later. Now we need to build to VAEs, the first one will reconstruct sentences from domainD1 and the second will reconstruct the sentences from domain D2. However, here we want the hidden state of VAE D1toD1 is the latent parallel sentence from domain D2. So, it needs to share a same distribution of the language model in domain D2. The figure here represents this theory. And I will implement this kind of method in my work.

Model implementation

We have a brief Introduction of the VAE method, next comes some more details. In my work, this should be VAE5-7-5 and VAE 7-5-7. Let's say the model5-7 is the encoder of the VAE 5-7-5. Consider the Decoder part of VAE 7-5-7 is also a 5-7 model. So we can share the same parameters. For simplicity, we just call a same model a model5-7 here. Here in my implementation, the model 5-7 is a sequence to sequence LSTM model and model 7-5 the same seq2seq thing. So actually. In this structure. The all we need Is two seq2seq model to reconstruct our input.

$$\begin{aligned}
&\log p(X, Y; \theta_{x|\bar{y}}, \theta_{y|\bar{x}}) \\
&\geq \mathcal{L}_{\text{ELBO}}(X, Y; \theta_{x|\bar{y}}, \theta_{y|\bar{x}}, \phi_{\bar{x}|y}, \phi_{\bar{y}|x}) \\
&= \sum\nolimits_i \Big[\mathbb{E}_{q(\bar{y}|x^{(i)};\phi_{\bar{y}|x})}[\log p(x^{(i)}|\bar{y};\theta_{x|\bar{y}})] - D_{\text{KL}}\big(q(\bar{y}|x^{(i)};\phi_{\bar{y}|x})||p_{\mathcal{D}_2}(\bar{y})\big)\Big] \\
&+ \sum\nolimits_j \underbrace{\Big[\mathbb{E}_{q(\bar{x}|y^{(j)};\phi_{\bar{x}|y})}[\log p(y^{(j)}|\bar{x};\theta_{y|\bar{x}})]}_{\text{Reconstruction likelihood}} - \underbrace{D_{\text{KL}}\big(q(\bar{x}|y^{(j)};\phi_{\bar{x}|y})||p_{\mathcal{D}_1}(\bar{x})\big)\Big]}_{\text{KL regularizer}}
\end{aligned}$$

The core part here is how to our loss function: the loss of VAE comes from two parts, first is the Reconstruction loss, which can be implement as the cross entropy of the reconstructed X with the target X and the KL divergence is the KL of the posterior $Q(\hat{y}|x)$ and prior $P(\hat{y})$ that is our language model. The reconstruction loss is easy to understand and the reaming part is how to compute the KL loss that comes with an expectation?

The answer here is we can not avoid from sampling, we need to sample some sentences from posterior $Q(\hat{y}|x)$ (For simplicity I'll just call it Q in the next). Then compute the log probability of this sentence in Q distribution which should be the entropy of Q when we go with expectation and the log probability of this sentence in language model distribution which is the cross entropy (Q, P) when we do go with expectation. The sample process which obeys Q distribution will theoretically lead us to the expectation over Q distribution.

Technical approaches

Gumbel-softmax

Now we know that the sample process over Q distribution will theoretically lead us to the expectation over Q and compute the KL divergence. So in real implementation of our deep learning model. How do we sample? A typical sample method does not come with gradient so we can not do backpropagation before the sample process. Here we need to use gumbel soft max trick: there is for a logits output vector of dimension d and we sample a same dimension gumbel vector that each value comes from a gumbel distribution. Then we sum the two vectors and softmax them. The probability after softmax is what we called the softoutput of gumbel - softmax. The soft output has gradient . Also we will compute a one hot vector based on the softmax output as the hard output of the gumbel-softmax. However we detach gradient of the difference between the hard part and the soft part.

In my implementation , the hard output is the latent parallel sentence, which is the output of my seq2seq model. This process is only used in the encoding process. Let's give an example here, for VAE 575, we model 57 will use gumbel softmax to generate the latent 7 sentence. And the model75 will use the argmax method since we do not need gradient after decoding and I also implement teacher forcing for training more easily in the decoding model.

Also we will notice that here for a datapoint step, we can sample from Q for K times and compute a average difference of those two log probabilities

Stop gradient.

This method is a empirical rule that proposed by the paper *'However, we find that approximating gradients of the reconstruction loss is much more challenging – both the Gumbel-softmax estimator and REINFORCE are unable to outperform a simple stop-gradient method that does not back-propagate the gradient of the latent sequence to the inference network.'* Note that although we stop gradient, the inference networks still receive gradients from the prior through the KL term, and their parameters are shared with the decoders which do receive gradients from reconstruction.

Let's take a example, for the cycle of model VAE 5-7-5 the parameters of model 57 are only updated by the KL loss by computing the log probability of a sentence in Q distribution and language model P distribution. And the reconstruction loss only optimize the parameters from model75. Also, in the VAE 757 cycle, the parameters of model 57 will be optimized by the reconstruction loss and parameters for 75 will be optimized by KL loss

Details about my implementation

Data

My data comes from GitHub and is gathered by Jackey Gao in traditional Chinese characters, thanks to his work. There are some characters that remain error codes in the raw data. My work is just deleting the whole sentence when we meet this kind of error. After split poems into sentences and data cleaning. I still have 614,000 five-word sentences and 546,000 seven-word sentences.

Code of gumbel-softmax

For gumbel-softmax code comes from the original paper which can be easily found on GitHub. I did not use the gumbel-softmax in Pytorch directly

Diminishing entropy of Q

Let's go back to our loss function, the KL function is minus entropy of Q plus cross entropy QP, so what we want is a higher entropy and a lower cross entropy during our training process. However, if you use the recons loss + kl loss directly, you will find the entropy of Q is decreasing however slower than the decreasing of cross entropy. So, the total KL loss decrease. However, the entropy of Q will decrease to zero after many steps. Which means the Q distribution only output a small space of sentences. And the model collapsed. In order to deal with it I enhance the ratio of reconstruction loss compared to the KL loss. Also we can change the ratio between entropy Q and cross entropy (Q, P). Then the entropy of Q will stop decreasing at about the number of 5 to 6 , which is a reasonable number for a distribution.

Posterior Collapse

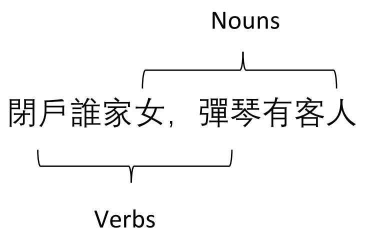
What I have fixed is that the posterior q distribution will not be like to centralized in a small space and have good variation. However, here comes a second hard problem. When I tried to sampling my final result from 5-word sentence to the latent seven-word sentence. I found that no matter what the input 5-word sequence. The posterior Q distribution will be like the same. That means if now I want to do the transduction from 5 to seven. The translation result will not depend on what 5-word sentence I have input.

A terminology for this is posterior collapse. when signal from input x to posterior parameters is either **too weak** or **too noisy**, and as a result, decoder starts ignoring z samples drawn from the posterior Which is a common problem for VAE.

That means the posterior $Q(\hat{y}|x)$ does not depend on x and my conditional language model deteriorate to a language model. In this case, the posterior Q just converge to the language model and ignore the input X. That means, the best case would be: my posterior Q have learned the position of common and stop, also my posterior can generate some sentences that not so good as the language model but is more or less acceptable. However, my task failed, that the generative sentence shares no meaning of the input sentence. Rather The attention structure in seq2seq model can not help with this

Also, this posterior collapse can be checked from my reconstruction loss. Since the latent sentences are about randomly generate sentences. The reconstruction model, who use argmax to sample sentence, can only output a small set of sentences. Which lead to the decreasing of my reconstruction loss after just 1-2 epochs. Here if I use some probabilistic sample to compute the reconstruction loss. I believe it's the same story: The difference will come from entropy of Q, not the difference between input x.

Demo from two language models



For both my two language models, when sampling , they 100% learned the position of comma and stop. The position information is easy for LSTM to learn. Also for the first 100 samples I found some amazing ones, which is almost the same to a human written poem sentence. Here the sample means *: Don't know who's girl inside, playing guitar with some guest.*

There are two pairs of words between the first half of sentence and second half of sentence that correspondent with their pos tags. And for rest of samples , some are not so fluent human like sentences but they mostly can match with pos tags. Also, in test set, my perplex of model is about 50, a reasonable number. So a simple language model works for the poem domain.

Some failed demos of my transduction model

The input five-word sentence x:

何以念兄弟，應思潔膳同。

The input five can be translated like: *How to miss my brother, imagine we drink eat together.*

Transduction result from probabilistic sampling:

 羡山有初令葱篇，日原垂捲十入耐。

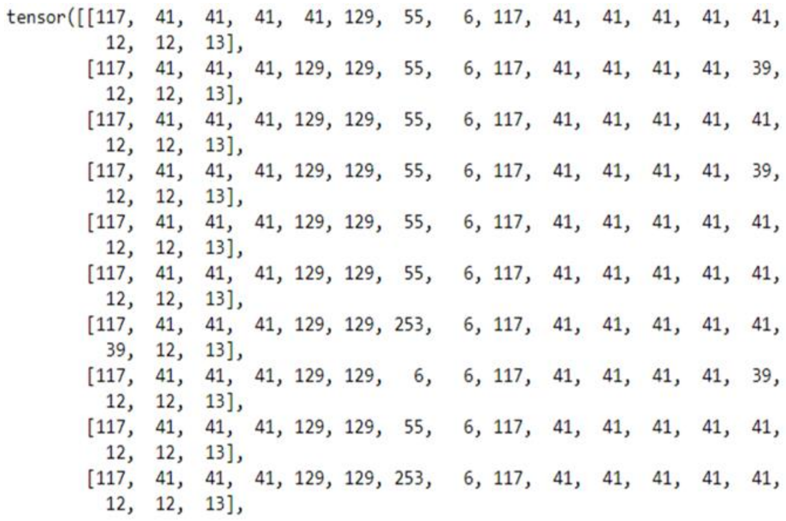
I don't know what is this sentence talking about , maybe some views. Translation directly would be: Envy the mountain has first green scallion, grassland enduring tiredness ten times at dusk.

庭有計豈曲衝孥，君五桐他燕無人

Translation directly: Writing song about leaving child in pavilion, gentleman go with sycamore he go with no one.

Not a fluent sentence, but mentioned the word gentleman, he and person, good to see.

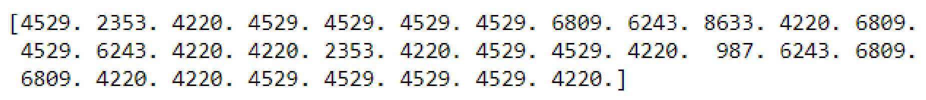
However, when you sample with argmax, which we suppose should be different among different X, the model just leave us some same output. Below are outputs from 10 different inputs x



```
tensor([[117,  41,  41,  41,  41, 129,  55,   6, 117,  41,  41,  41,  41,  41,
          12,  12,  13],
        [117,  41,  41,  41, 129, 129,  55,   6, 117,  41,  41,  41,  41,  39,
          12,  12,  13],
        [117,  41,  41,  41, 129, 129,  55,   6, 117,  41,  41,  41,  41,  41,
          12,  12,  13],
        [117,  41,  41,  41, 129, 129,  55,   6, 117,  41,  41,  41,  41,  39,
          12,  12,  13],
        [117,  41,  41,  41, 129, 129,  55,   6, 117,  41,  41,  41,  41,  41,
          12,  12,  13],
        [117,  41,  41,  41, 129, 129,  55,   6, 117,  41,  41,  41,  41,  41,
          12,  12,  13],
        [117,  41,  41,  41, 129, 129, 253,   6, 117,  41,  41,  41,  41,  41,
          39,  12,  13],
        [117,  41,  41,  41, 129, 129,   6,   6, 117,  41,  41,  41,  41,  39,
          12,  12,  13],
        [117,  41,  41,  41, 129, 129,  55,   6, 117,  41,  41,  41,  41,  41,
          12,  12,  13],
        [117,  41,  41,  41, 129, 129, 253,   6, 117,  41,  41,  41,  41,  41,
          12,  12,  13],
```

Reconsider the problem

Finally, I dig deeper into my seq2seq model. Something surprised me and I might found the reason for posterior collapse.



```
[4529. 2353. 4220. 4529. 4529. 4529. 4529. 6809. 6243. 8633. 4220. 6809.
 4529. 6243. 4220. 4220. 2353. 4220. 4529. 4529. 4220.  987. 6243. 6809.
 6809. 4220. 4220. 4529. 4529. 4529. 4529. 4220.]
```

Output of the first decoded word index when I inputs 32 different inputs to a initialized Seq2seq model

When you initialize a seq2seq LSTM model, the first decoding word with different 5 sentence inputs have already be almost the same. I have a batch size of 32 which means the inputs are different 32 sentence. However, the initialized parameter will ignore the different sentences and gives you the same encoded hidden state. I've tried it many times. And not only seq2seq model. For A vanilla LSTM language model, when you initialize it. It will be the same situation. So that means at the first initialize step, the model are likely to output a small space. if this is a labeled seq2seq LSTM model, the model will learn to expand the output space in order to fit the cross-entropy loss. However, for my implementation, we do not have the label data for seq 2 seq output and the gradient stopped after this is also a problem. There is not much penalty for the same posterior $Q(\hat{y}|x)$ in this VAE loss function. Thus, the model learnt to output the same argmax choice all the time. The model just learned to output a same Q posterior which will lead to the failure in reconstruction loss but can not find a good gradient to do better reconstruction work.

$$\mathcal{L}_{\text{rec}} = -\alpha \cdot \sum\nolimits_i [p_{\text{dec}}(e(x^{(i)}), c_x)] - \alpha \cdot \sum\nolimits_j [p_{\text{dec}}(e(y^{(j)}), c_y)]$$

A proposed way to fix this problem, add a third loss: At the first several epochs, inputs the 5-word sentence into the model5-7 and use the 5 word sentence it self as the target. Also do the same thing for the model7-5. This enhances the initialization parameters. And force the seq2seq model to output a different posterior when inputs are different. And the reconstruction loss of this 5-5 and 7-7 should be considered in the first several epochs. Also you can set a scheduler for the ratio of this loss or some more detailed implementations. due to time limitation. I did not try this method. But I hope it will work.