# Generating Semantic Sentences

Manish Periwal[1]

***Abstract*— Given the growing complexity of tasks and data in NLP there are limited unsupervised learning methods to tackle the problems. In this paper we will look into approach to one of the problems as representing sentences in latent space using Recurrent Neural Network(RNN) and Variational Auto Encoder. Combining VAE-LSTM approach we will repharase (and generate similar and meaningful) sentences from given sentence. Although we assumption same architecture can also be applied to language modeling problem. We will give results on using single layer encoder as well as 2 layer encoder for our VAE.**

## I. INTRODUCTION

The aim of an autoencoder as Mutilayer Perceptron(MLP) is to learn a representation (encoding) for a set of data, typically for the purpose of dimensionality reduction. In recent years, the autoencoder especially Variational Autoencoder(VAE, Kingma et al., 2015 and Rezende et al., 2014) concept has become more widely used for learning generative models of data. VAE are interesting and one of the most efficient method for unsupervised learning as generative models, which combine ideas from deep learning with statistical inference. They can be used to learn a low dimensional representation Z of high dimensional data X such as images (of e.g. faces) or NLP (of e.g sentences). VAEs have already shown promise in generating many kinds of complicated data, including handwritten digits (Diederik et al., 2014 and Tim et al., 2015), faces (Tejas et al., 2014 and Danilo et al., 2014), house numbers (Diederik et al., 2014 and Karol et al., 2015), CIFAR images (Karol et al., 2015), physical models of scenes (Tejas et al., 2015), segmentation (Kihyuk et al., 2015), and predicting the future from static images (Jacob et al., 2016). RNN(Recurrent Neural Networks) with great success has also been used at industrial level(machine translation) for mappping sequence to sequence(time series) data. An MLP can only map from input to output vectors, whereas an RNN can in principle map from the entire history of previous inputs to output. RNN also helps the network learn the language modeling. In this paper we will learn to generate similar contextual sentences given a sentence with same using VAE RNN.

Two other models have shown promise in learning sentence encodings, but cannot be used in a generative setting: Skip-thought models (Kiros et al., 2015) are unsupervised learning models that take the same model structure as a sequence autoencoder, but generate text conditioned on a neighboring sentence from the target text, instead of on the target sentence itself. Finally, paragraph vector models (Mikolov et al., 2014) are non-recurrent sentence representation models. In a paragraph vector model, the encoding of a sentence is obtained by performing gradient-based inference on a prospective encoding vector with the goal of using it to predict the words in the sentence.

While vaes have showed good results on continous domains like images, two similar model is being used at discrete sequences: a technique for doing this using rnn encoders and decoders VRAE (Mikolov et al., 2014) and Generating Sentences from a Continuous Space (Oriol et al., 2016. Though these models did not showcase the reults of using multilayer Neural Network each of encoder and decoder which we will cover as well.

## II. BACKGROUND

### A. LSTM

Long Short-Term Memory (LSTM) introduced by Hochreiter & Schmidhuber (in 1997) is a specific recurrent neural network (RNN) architecture that was designed to model temporal sequences and their long-range dependencies more accurately than conventional RNNs. In this paper we use LSTM as encoder for converting high dimensional textual data to low dimension latent space and decoder for mapping latent space to generating sentences.

An LSTM network computes a mapping from an input sequence x = (x1, ..., xT ) to an output sequence y = (y1, ..., yT ) by calculating the network unit activations using the following equations iteratively from t = 1 to T:

$$i_t = \sigma(W_{ix}x_t + W_{im}m_{t1} + W_{ic}c_{t1} + b_i) \quad (1)$$

$$f_t = \sigma(W_{fx}x_t + W_{fm}m_{t1} + W_{fc}c_{t1} + b_f) \quad (2)$$

$$c_t = f_t \odot c_{t1} + i_t \odot g(W_{cx}x_t + W_{cm}m_{t1} + b_c) \quad (3)$$

$$o_t = \sigma(W_{ox}x_t + W_{om}m_{t1} + W_{oc}c_t + b_o) \quad (4)$$

$$m_t = ot \odot h(c_t) \quad (5)$$

$$y_t = \varphi(W_{ym}m_t + by) \quad (6)$$

where the W terms denote weight matrices (e.g. $W_{ix}$ is the matrix of weights from the input gate to the input), $W_{ic}$, $W_{fc}$, $W_{oc}$ are diagonal weight matrices for peephole connections, the b terms denote bias vectors ($b_i$ is the input gate bias vector), $\sigma$ is the logistic sigmoid function, and i, f, o and c are respectively the input gate, forget gate, output gate and cell activation vectors, all of which are the same size as the cell output activation vector m, $\odot$ is the element-wise product of the vectors, g and h are the cell input and cell output activation functions, generally and in this paper tanh,

[1]Email - manishperiwal2009@gmail.com

and $\varphi$ is the network output activation function, softmax in this paper.

### B. Variational AutoEncoder (VAE)

In the probability model framework, a variational autoencoder contains a specific probability model of data xx and latent variables zz. A VAE consists of an encoder, a decoder, and a loss function. In probability model terms, the variational autoencoder refers to approximate inference in a latent Gaussian model where the approximate posterior and model likelihood are parametrized by neural nets (the inference and generative networks).

The encoder is a neural network that outputs a representation zz of data xx which can be inference network in probablity model terms and parametrizes the approximate posterior of the latent variables zz. The inference network outputs parameters to the distribution $q(z \mid x)$.

The decoder is a neural net that learns to reconstruct the data xx given a representation zz which again acts as generative network in probablity model. The likelihood of the data xx given latent variables zz is parametrized by a generative network. The generative network outputs parameters to the likelihood distribution $p(x \mid z)$.

## III. Our Model

The key to lstm is cell state. The lstm have the ability to remove or add information to the cell state, carefully regulated by structures called gates. We used single layer encoder and 2 layer encoder but found out that single layer gives better results if timesteps are less. In single layer encoder the state of encoder is used to produce latent space. As we also tried 2 layer encoder, in which case we would take the state of last layer of lstm as input to our vae and produce latent space. We would then pass the latent space into decoder(which is another single layer lstm) and produce similar sentences.

Our loss function at the decoder would be

$$L(\theta; x) = KL(q(z \mid x) \mid\mid p(z)) + E_{q_\theta(z|x)}[\log p(x \mid z)] \leq \log p(x). \tag{7}$$

where KL-Divergence $\geq 0$ depends on how good $q(z \mid x)$ can approximate $p(z \mid x)$.

### A. Points to remember

- Training is tricky. Vanilla training results in the decoder ignoring the encoder and the KL error term becoming zero.
- Word dropout using a word keep rate hyperparameter. This forces the decoder to rely more on the global representation.

## IV. Experiments

### A. Data and preprocessing:

For our experiments we used open source cornell movie dialogue and tv series Friends corpus. We constarined on sentences less than or equal to 10 timesteps(words).

### B. Training a model:

We started with vanilla sequence to sequence for generation of sentences and as expected output are just copy of input sentences. We then started adding a layer autoencoder which acts as latent space for input nevertheless the output were more or less copy of input. Gradually we then moved to following architecture for generating sentences. Table I, II and III show the results from different architecture for generating sentences.

We converted each sentence(dialogue) into 10 words(time steps). These words were passed to rnn embedding layer which would convert each word into 128 dimension vector. We also tried replacing rnn embedding layer to using predefined word2vec but the results were almost same. The vectors from embedding layer were passed to 1 layer and 2 layer lstm encoder independently. We composed the state size of each cell to 128 dimension vector and each layer have 128 lstm cell units in encoder. The state was further projected into latent varaibles. We experimented with different size of latent variable from 5 to 10 but no significant improvement was observed in changing the size. The decoder was again single layer with 128 lstm units in a layer. We also experimented with different dropout in encoder and decoder. We used AdamOptimizer as optimzer and clipped gradients to 5.

| Input Sentence | Generated Sentence |
|---|---|
| this is just something i have to do | this is just really i have to do |
| can i sing at your wedding ? | can i ll to your wedding ? |
| central perk, joey and phoebe are talking | central perk, phoebe and phoebe are entering |
| i i got a date tonight | i got a date too |

TABLE I
VANILLA VAE-LSTM WITHOUT WORD DROPOUT

| Input Sentence | Generated Sentence |
|---|---|
| what ' s it say ? | what ' s going talking ? |
| you must be so happy ! | you got be so great ! |
| he glares at chandler | she looks at joey |
| oh , how ' d she get back ? | oh , how ' d you going back ? |

TABLE II
1 LAYER ENCODER VAE-LSTM WITH WORD DROPOUT

| Input Sentence | Generated Sentence |
|---|---|
| that was the word i was looking for | it was the way i was going for |
| you ' re the guys from the movie ! | you ' re the little in the life ? |
| and because they have warp capabilities , the consequences to their society are minimal | and if i have to and , he hell is |
| ok ok we ' re just gonna head inside here | no , i ' re not going get in ! |

TABLE III
2 LAYER ENCODER VAE-LSTM WITH WORD DROPOUT WITH MAX 15 WORDS

## V. CONCLUSIONS

LSTM helps network remember long term dependecies and VAE can very efficiently compress high dimension input text to latent variables. We prove that combining VAE and LSTM can produce semantic similar sentences. In future work we hope to investigate and traverse into latent space and generate sentences based on context as well by making network deep(multi layer encoder decoder). We would also like to generate longer semantic sentences in which case increasing number of layers might help.

## REFERENCES


[1] Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. ICLR, 2014.

[2] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochas-tic backpropagation and approximate inference in deep generative models. In ICML, 2014.

[3] Tim Salimans, Diederik Kingma, and Max Welling. Markov chain monte carlo and variational inference: Bridging the gap. ICML, 2015.

[4] Tejas D Kulkarni, William F. Whitney, Pushmeet Kohli, and Josh Tenen- baum. Deep convolutional inverse graphics network. In NIPS, 2015.

[5] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Rezende, and Daan Wierstra. Draw: A recurrent neural network for image generation. In ICCV, 2015.

[6] Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In NIPS, 2014.

[7] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In NIPS, 2015.

[8] Jacob Walker, Carl Doersch, Abhinav Gupta, and Martial Hebert. An uncertain future: Forecasting from static images using variational autoencoders. In ECCV, 2016.

[9] Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-thought vectors. arXiv preprint arXiv:1506.06726.

[10] Quoc V. Le and Tom as Mikolov. 2014. Distributed representations of sentences and documents. In Proc. ICML.

[11] Otto Fabius & Joost R. van Amersfoort 2015.VARIATIONAL RECURRENT AUTO-ENCODERS. arXiv preprint arXiv:1412.6581v6.

[12] Oriol Vinyals, Andrew M. Dai, Rafal Jozefowicz & Samy Bengio, Samuel R. Bowman, Luke Vilnis 2016. Generating Sentences from a Continuous Space. arXiv preprint arXiv:1511.06349.

[13] Hochreiter, Sepp and Schmidhuber, Jurgen. Long short-term memory. Neural computation, 9(8):17351780, 1997.

[14] Diederik P. Kingma and Max Welling. 2015. Auto-encoding variational bayes. In Proc. ICLR.

[15] Danilo J. Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. In Proc. ICML.