# UROP Report – Neural Dialogue System with Diversity

Contribution: Fang Haoyang (50%), Gao Tong (50%)

## Abstract

Neural dialog system has been an active area of research. Most neural dialog systems follow the sequence-to-sequence framework, that is, to generate an answer for a given question. However, traditional models have a 1-to-1 assumption between questions and answers and neglect the distributions of the target sequences, thus producing monotonous answers. In our research project, we apply a variational sequence to sequence model to generate diverse answers. First, we introduce the mathematical background. Second, we implement DIAL-LV [3], a model proposed recently to generate diverse answers. Finally, we evaluate the performance of the implemented model.

## Introduction

Generative models were popular in image processing. In particular, the Variational Auto-Encoder (VAE) [5] shows a potential effect on generative models in text processing field, and then the latent variable trick dominates recent publications [2, 3, 6, 7, 8]. VAE solves two essential problems: the definition of latent variable and how to integrate over it [4]. the focus of following papers moved on to the concept and structure of the model.

Google Brain introduces a model that describes the latent distribution of the entire sentences [2]. It improves the standard RNN language model, which generates words one by one, lacking global features. Under the influence of this paper, more concepts and structures of models appeared, for example, a new generative neural variational framework [7], a discrete model for sentence compression [6], a model describing the answer distribution as well as generating diverse answers [3], etc.

While all the models used variational Bayesian methods, we first introduce the mathematical concepts of the models. Then, we analyze the ideas behind the structures of the models and introduce tricks to deal with the confronting problems. Finally, we implement and evaluate the latest model.

## Mathematical Background

1. Maximum a posteriori estimation (MAP):

   MAP is the fundamental estimation in Bayesian statistics where we use a prior distribution g over θ (θ is the parameter of our model and it's called Maximum Likelihood Estimation (MLE) when g is trivial), and use

   $$\theta \mapsto f(\theta \mid x) = \frac{f(x \mid \theta)\, g(\theta)}{\displaystyle\int_{\vartheta \in \Theta} f(x \mid \vartheta)\, g(\vartheta)\, d\vartheta}$$

   to calculate the posterior distribution of θ. Then MAP gives the estimation of θ by

   $$\hat{\theta}_{\mathrm{MAP}}(x) = \underset{\theta}{\arg\max}\, f(\theta \mid x) = \underset{\theta}{\arg\max}\, \frac{f(x \mid \theta)\, g(\theta)}{\displaystyle\int_{\vartheta} f(x \mid \vartheta)\, g(\vartheta)\, d\vartheta} = \underset{\theta}{\arg\max}\, f(x \mid \theta)\, g(\theta).$$

   However, when a latent variable z is introduced, we have

   $$p(x|\theta) = \int p(z|\theta)p(x|z,\theta)\, dz,$$

   and if this is not differentiable or cannot be evaluated, we are not able to use MAP directly.

2. Expectation–maximization algorithm (EM algorithm):

For problems with latent variables introduced, usually it’s not possible to marginalize the latent variable and to optimize the objective directly by MAP/MLE. EM algorithm, an iterative method to estimate the value according to MAP/MLE, is born to solve this problem. Due to space restriction, the specific description and proof is omitted.

### 3. Variational Bayes Inference:

However, because of the intractability of latent variable, both MAP/MLE and EM algorithm cannot be implemented in most cases in our learning process. Considering we always deal with large datasets, Monte Carlo implementations are not feasible as well. So we need a further improvement for EM algorithm, and that is variational Bayes inference (VB) [1].

There are two main purpose of VB:

1) Estimate the posterior distribution of the latent variable so that we can apply statistical inference methods on them.
2) For a specific model, generate an evidence lower bound (ELBO), which is used for model selection.

VB is widely used in deep learning area and both models we are going to talk about used VB to do evaluations.

## Model Description (LVDM)

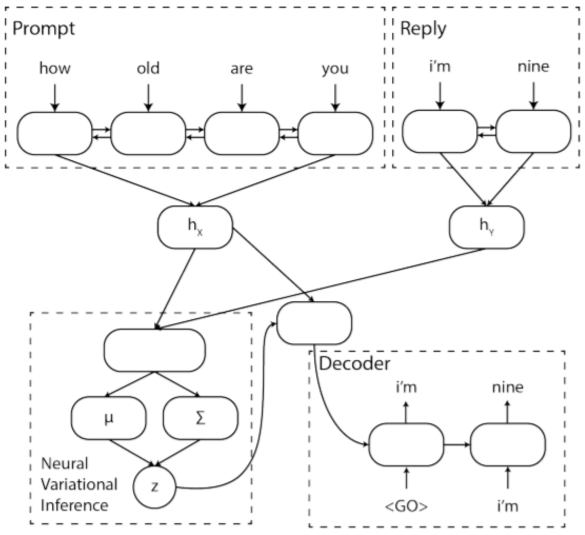

Figure 1: the schematic of LVDM (For Training)

For higher diversity in dialogue response, Latent Variable Dialogue Model (see Figure 1) introduced a latent variable z with a standard Gaussian prior in decoding process. Traditionally, encoder-decoder model will encode all possible replies to X in decoder's probability distribution $P(Y|X)$ and makes it hard to disentangle possible replies. Now with the help of stochastic component z, the replies are stored in the probability distribution $P(Y|z,X)$ rather than $P(Y|X)$ and no exact maximum likelihood search can be performed for reply Y, thus we can avoid boring reply problem.

When training, the model first feeds prompt X and reply Y into two independent bidirectional RNNs with GRU cells, and get hx and hy from their hidden states passed through a single nonlinear layer. Then it calculates mean and variance of proposal distribution $Q(z|X,Y)$ as:

$$\mu = W_\mu[h_\mathbf{x}\ h_\mathbf{y}] + b_\mu$$
$$\log(\Sigma) = diag(W_\Sigma[h_\mathbf{x}\ h_\mathbf{y}] + b_\Sigma)$$

where $Q(z|X,Y)$ is an approximation of the posterior $P(z|X,Y)$, [a b] denotes the concatenation of a and b, and diag denotes inserting along the diagonal of a matrix.

With distribution $Q(z|X,Y)$, we can sample z using a reparametrization trick, initialize the hidden state of decoder GRU with [hx z] and train it by minimizing the following ELBO for the log-likelihood of the data:

$$\log P(Y|X) \geq -\mathcal{KL}(Q(z|X,Y)||P(z)) \\ + \mathbb{E}_{z\sim Q}\log P(Y|z,X)$$

There is a slight different in training and predicting of this model. At generation time the latent variable z is sampled from prior distribution $N(0, I_n)$ and replies is decoded from $P(Y|z,X)$.

## Tricks:

1. Annealing KL Term:

   As RNNs are powerful density estimators, the model tends to force the KL term to 0 and optimize the data reconstruction term of the ELBO, which leads to the failure in encoding useful information into z. The paper pointed out that the problem can be solved by gradually annealing the KL term weight over the course of model. In practice, we used a piecewise function f(x) to calculate the KL term weight as:

$$\mathrm{f(x)} = \begin{cases} 0 \;\; (0 \leq x < 0.1t) \\ \dfrac{\mathrm{m}(x-0.1t)}{0.8t} \;\; (0.1t \leq x < 0.9t) \\ m \;\; (0.9t \leq x \leq t) \end{cases}$$

   where x denotes current time step, m denotes the maximum weight of KL term, and t denotes target training times.

   However, we found another problem in KL term during training. In practice, after a few loops of training, the KL term will be too small for system to represent and finally become NAN, which stops the system from training. We solved it by setting up a lower bound for KL term so that further trial to minimize the extreme small KL term is prohibited. In our implementation, the lower bound is set to 0.0002.

2. Reparametrization Trick:

   In order to generate sample from distribution $Q(z|X, Y)$, this trick as to express the random variable z as a deterministic variable is introduced [5], thus a differential estimator can be constructed. In this case, as z follows a Gaussian distribution, we can sample $z \sim N(\mu, \Sigma)$ by first choosing an $\epsilon$ from standard distribution $N(0, I_n)$ and then let $z = \mu\epsilon + \Sigma$. By implementing this trick, a differentiable estimator of variational lower bound can be obtained and the whole training process can run smoothly.

## Implementation (LVDM)

1. Preprocess:

   First, we convert all QA pairs into integers and save them into tfrecords for better training performance. Then, we split questions and answers, strip all punctuations, add <GO> and <EOS> symbols to all sentences, and used collections module in python to convert words to integers efficiently. One severe problem we face is that our primary implementation is overusing all the memory because the program simply loaded all data into memory without any optimization. We fix this problem by loading dataset line by line and utilizing the update function of python Counter to filter out words with low occurrence rate every mini-batch, by which memory use and processing time are reduced significantly. The read_and_decode function in reader.py is also implemented for loading data from tfrecords in training and prediction process.

## 2. Main:

In main.py we constructed the model graph in tensorflow and implemented training and prediction function. We first found the paper ambiguous, because the model described in this paper can be only applied to the case that both prompt and reply are given, which means that it cannot be used for prediction. We then asked the author through email and got quick response from him. According to his reply, we made $z \sim N(0, I_n)$ when predicting and the whole model became clear.

Next problem is the concrete representation of the second term in ELBO. Since both paper [3, 5] lack detailed explanation, we had to figure it out by ourselves. We made an analogy to seq2seq model and used the tf.nn.softmax_cross_entropy_with_logits to represent it, which is proven to be reasonable from the training result.

The last technical problem is that tensorflow does not support the feed_previous function in their RNN implementation, therefore we implemented a RNN with feed_previous function using raw_rnn in tensorflow, and it worked well during testing. Note that for training we used dynamic RNN to simplify training process, and that's why the training and prediction processes used slightly different RNN with same internal parameters.

## 3. Evaluation:

a. We first calculate the frequency of each word and select the top 20 words. Then, we plot the graph with logrithm of the frequency rank of the words as x-axis and logrithm of frequency of the words as y-axis. We use linear regression to find parameter k, b in the linear equation y = kx + b, and use -k as the estimation of the Zipf parameter.

b. We implement uniqueness check in two ways. First we find the uniqueness rate ( $\text{answers}_{\text{unique}}/\text{answers}_{\text{total}}$ ) for each question, and calculate the mean of rates of all answers as the result. This uniqueness is used to check the effectiveness of the latent variable. Second we calculate the rate of unique ones among all answers ( $\text{answers}_{\text{unique}}/\text{answers}_{\text{total}}$ ). This uniqueness is used to check the simple answer problem --- for instance, “I don’t know” is a reasonable answer for every question but we don’t really like it to appear many times.
c. We also record the KL term loss, expectation loss and total loss after training and use them for evaluation as well as comparison with other models.

## Conclusion

In this project, we implement a variational sequence-to-sequence model for the dialogue system. We first present some mathematical toolsfor the models. Then, we implement the LVDM model, which estimates the distribution of answers and generates diverse answers. Finally, we evaluate the performance of the implemented model.

## References

[1] Beal, M. J. (2003). Variational algorithms for approximate bayesian inference. *University College London.*

[2] Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Jozefowicz, R., & Bengio, S. (2016). Generating sentences from a continuous space. *Conference on Computational Natural Language Learning (CoNLL).*

[3] Cao, K., & Clark, S. (2017). Latent Variable Dialogue Models and their Diversity. *European Chapter of the Association for Computational Linguistics (EACL).*

[4] Doersch, C. (2016). Tutorial on variational autoencoders. Retrieved from: https://arxiv.org/abs/1606.05908

[5] Kingma, D. P., & Welling, M. (2014). Auto-Encoding Variational Bayes. *International Conference on Learning Representations (ICLR).*

[6] Miao, Y., & Blunsom, P. (2016). Language as a Latent Variable: Discrete Generative Models for Sentence Compression. *Conference on Empirical Methods in Natural Language Processing (EMNLP).*

[7] Miao, Y., Yu, L., & Blunsom, P. (2016). Neural variational inference for text processing. *International Conference on Machine Learning (ICML).*

[8] Serban, I. V., Sordoni, A., Lowe, R., Charlin, L., Pineau, J., & Courville, A., et al. (2016). A hierarchical latent variable encoder-decoder model for generating dialogues.