Deep Latent Variable Models for Text Generation

Tushar Goel
École Polytechnique Fédérale de Lausanne, Switzerland

I. Abstract

The ever increasing growth in the usage of Social Media and the E-Commerce has led to a burgeoning volume of textual data. This growth has resulted in the need for efficient processing of these text corpuses. The aim of this project is to find smooth and continous latent representations for language processing which can be used to generate and summarize text. We begin by exploring the performance of the state of the art Sequence to Sequence Deep Learning models and then switch to generative and reinforcement learning based methods to perform this task.

II. Introduction

The goal of this project is to find smooth and continous latent representations of natural language. These smooth and continous latent representations can then be used for further downstream tasks like text summarization, Question/Answering, Chatbots, etc. Hence, in this project, we will be trying to pass text sentences through an encoder to generate a fixed size latent space and then try to reconstruct the entire sentence from those latent spaces by passing these encoded variables through a decoder. The quality of these reconstructions will be judged by evaluation metrics like BLEU and ROUGE scores which are explained later.

In the past, several approaches have shown that sequence to sequence LSTM based models work quite nicely for text based Variational AutoEncoders([1]). We try to compare the performance of these standard LSTM based Encoder to Decoder Models with Pre-trained Language Models to see if we can leverage their expressiveness provided by these Pre-trained Language Models.

In this work, we make use of Optimus([2]), a large-scale pre-trained deep latent variable models for natural language. OPTIMUS is composed of multi-layer Transformer-based encoder and decoder mechanism which enjoys several favorable properties as it combines the strengths of VAE([1]), BERT([3]) and GPT([4]), and supports both natural language understanding and generation tasks. As we can see in the figure of Optimus architecture, the encoder has the same configuration as BERT, the decoder has the same configuration as GPT2 and there is a 32 dimensional latent space connecting them.

The following loss objective is minimized by these encoder-decoder architectures:

$$L_{Optimus} = L_E + \beta L_R$$

$$L_E = -E_{q_\phi(z|x)} log[p_\theta(x|z)]$$

$$L_R = KL(q_\phi(z|x)||p(z))$$

Fig. 1: Optimus architecture

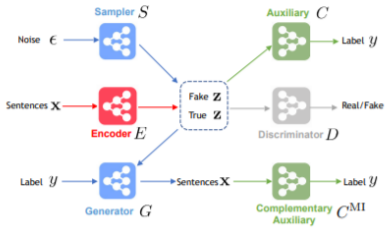

Fig. 2: CARA architecture

To further optimize our models, we make use of Complementary Auxiliary classifier Regularized Auto-encoder (CARA) ([5]) to learn latent sentence representations which are disentangled with the labels. CARA achieves this by adding two classifiers to predict labels: a classifier in the latent space is used for disentangling feature learning as in ARAE([6]), and a complementary classifier in the observation space is proposed to encourage the generator to contain the conditional label.

$$L_{rec} + L_{disentangle} + L_{adversarial}$$

$$min_{E,G}\ L_{rec} = -E_{q_E(z|x),p(y)}\ log\ p_G(x|z,y)$$

$$min_E\ max_C\ L_{disentangle} = E_{q_E(z|x)}\ log\ [p_C(y|z)]$$

$$min_{E,S}\ max_D L_{adversarial} = E_{x\sim p(x)}\ log\ p_D(d=1|E(x))$$

$$+E_\epsilon\ log\ p_D(d=0|S(\epsilon))$$

LSTM based decoder models, as we observe in our work, are prone to repeat the words which are decoded by the model. This effect is observed due to the problem of exposure bias([8]), since the models are trained on the ground truth data but have to predict words autoregressively during the test time. To overcome this, we try to incorporate the prox-policy optimization ([7]) training regime which is a Reinforcement Learning based algorithm where a reward function is maximized. In this method, we let the decoder generate the entire text and then we try to optimize the text for the desired evaluation metric. The method only updates the values of the decoder and has no effect on the encoder and should

be used only for fine-tuning. The following reward function is optimized by the PPO algorithm:

$$R(x, y) = r(x, y) - \beta \frac{log\ \pi(y, x)}{log\ \rho(y, x)}$$

Please note that x represents a prompt for the decoder, y represents the decoded value of the prompt. We elaborate later on how the function $r(x, y)$ is defined. It must be noted that π represents the base decoding policy and ρ represents the new policy. We add the extra term as a regularizer to prevent the policy from changing a lot from the initial policy.

III. Dataset and Methodology

In our work, we will use the Yelp Review dataset. This dataset contains around 500k reviews of different businesses like resteraunts and hotels. The following features are provided in the dataset:

- Text: The review of a business
- Votes: Number of votes given to the comment in the following settings: "useful", "funny" and "cool"
- Stars: The rating of a user for the business

The following is an outline of the strategy we have employed in our methodology:

- Tokenize and embed the reviews into both the BERT and GPT-2 Embeddings
- Train the Optimus Model to reconstruct the reviews with the latent layer
- Train the CARA model using the Optimus pre-trained weights. We will use votes as the labels for the reviews hence these labels will be integers from 1 to 5
- Use the weights of the best model and then fine-tune the PPO algorithm

IV. Experiments and Results

In this section we discuss the results obtained for the above defined methodology. We use Adam optimizer and the learning rate is set to $5X10^{-5}$ if not specified otherwise.

A. Fine-tuning on Optimus

We train our algorithms for $\beta = 0$ and $\beta = 0.5$. The size of the latent space is 32 bits. As we will show later, we notice that the models perform best for $\beta = 0.5$. During the training regime, we notice that while reconstruction loss is the minimum for $\beta = 0$, the KL divergence shoots up exponentially thereby leading to a worse overall performance

B. Training CARA

The CARA model is trained using the weights from the OPTIMUS model to which we add the two classifiers and generators. All the classifiers and generators added in CARA are linear transformations to the desired dimension space with a Softmax layer for the classifiers. Hence, the noise sampler S is a $32X1$ linear transformation, generator G is $5X32$ transformation and Auxilary classifier C is a $32X5$ linear transformation with softmax. What we notice while training

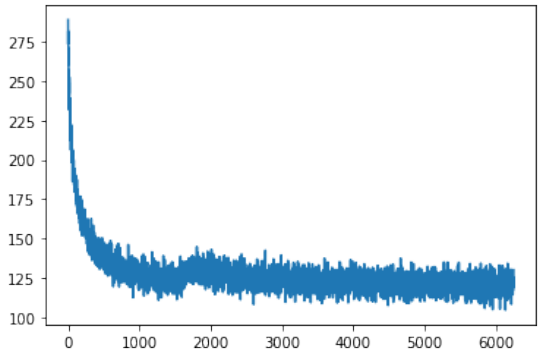

Fig. 3: Reconstruction loss for $\beta = 0.5$

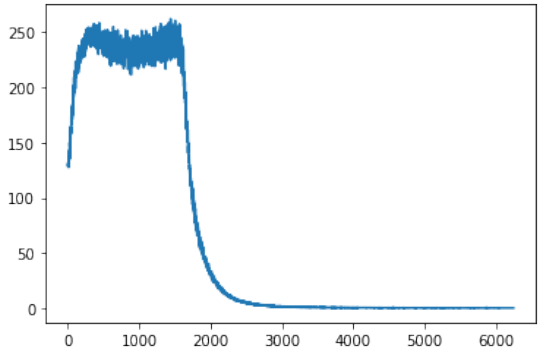

Fig. 4: KL divergence loss for $\beta = 0.5$

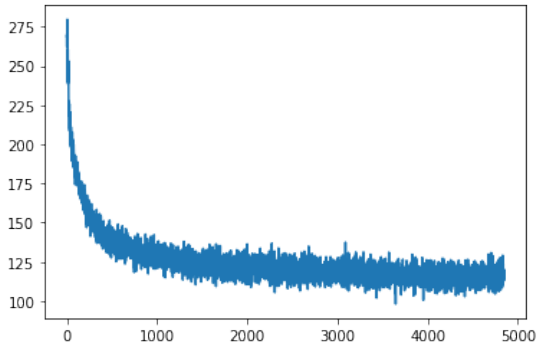

Fig. 5: Reconstruction loss for $\beta = 0$

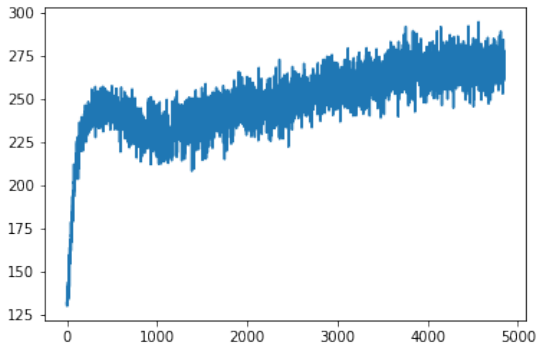

Fig. 6: KL divergence loss for $\beta = 0$

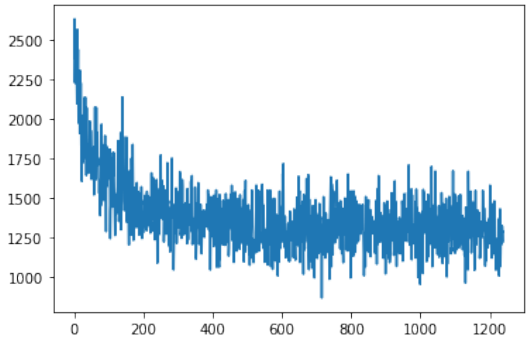

Fig. 7: Total loss for CARA

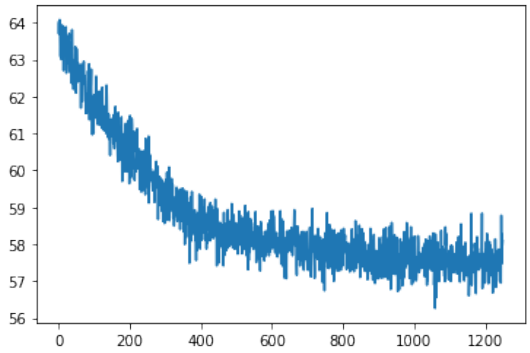

Fig. 8: Adversarial + Disentangled loss for CARA

is that the reconstruction loss in CARA improve slightly but the major improvement comes from the disentangled and adversarial loss. As we will see later, using CARA improves the quality of reconstruction slightly.

C. Training Prox Policy Optimization

We define the reward function $R(x, y)$ as the difference in the Bleu score obtained from the model currently being used with the Bleu score of the learnt model from the initial policy obtained from OPTIMUS. The difference is used because we want to give the model a baseline on top of which the performance must be improved. It must be noted that we use PPO as a fine-tuning algorithm on top of Optimus. The training implementations were kept the same as in the original paper([9]).

D. Test set Results

During the test time, we evaluate our models based on the BLEU and ROUGE scores on a test set of 1000 reviews. The following results were observed:

TABLE I: Performance of the Optimus Model for text generation

Model	Rouge-1	Rouge-2	Rouge-L	Bleu
LSTM	8.7	3.2	8.5	8.2
Optimus($\beta = 0$)	21.3	5.2	19.1	22.4
Optimus($\beta = 0.5$)	19.3	5.3	18.9	19.2
CARA	24.1	6.8	23.3	25.2
PPO	19.9	4.9	19.4	20.5

V. Conclusion

We can clearly see the huge improvement in using Pre-trained Language Models as compared to the standard LSTM based Encoder Decoder Mechanisms. We also notice an improvement in the reconstruction quality as we introduce an adversarial and a generative approach to training these pre-trained models. Unfortunately, we do not see any noticeable improvement in reconstruction quality as we use Reinforcement Learning based methods. This might be due to the fact that we do not have a good adaptive baseline for the RL models.

References

[1] Diederik P Kingma, Max Welling, *Auto-Encoding Variational Bayes*. In: Arxiv 2014

[2] Chunyuan Li, Xiang Gao, Yuan Li, Baolin Peng, Xiujun Li, Yizhe Zhang, Jianfeng Gao, *Optimus: Organizing Sentences via Pre-trained Modeling of a Latent Space*. In: EMNLP 2020

[3] Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* . In: Arxiv 2019

[4] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever *Language Models are Unsupervised Multitask Learners*. In: Arxiv 2018

[5] Yuan Li1, Chunyuan Li2, Yizhe Zhang, Xiujun Li, Guoqing Zheng, Lawrence Carin , Jianfeng Gao *Complementary Auxiliary Classifiers for Label-Conditional Text Generation*. In: Association for the Advancement of Artificial Intelligence 2020

[6] Jake (Junbo) Zhao, Yoon Kim, Kelly Zhang, Alexander M. Rush, Yann LeCun *Adversarially Regularized Autoencoders*. In: ICML 2018

[7] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, Oleg Klimov *Proximal Policy Optimization Algorithms*. In: Arxiv 2017

[8] Florian Schmidt *Generalization in Generation: A closer look at Exposure Bias* In: Proceedings of the 3rd Workshop on Neural Generation and Translation 2019

[9] Nisan Stiennon, Long Ouyang, Jeff Wu, Daniel M. Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, Paul Christiano *Learning to summarize from human feedback* In: Arxiv 2021