OPTIMUS: Organizing Sentences via Pre-trained Modeling of a Latent Space

Chunyuan Li, Xiang Gao, Yuan Li, Xiujun Li, Baolin Peng, Yizhe Zhang, Jianfeng Gao
Microsoft Research, Redmond
{chunyl, xiag, v-liyua, xiul, bapeng, yizzhang, jfgao}@microsoft.com

Abstract

When trained effectively, the Variational Autoencoder (VAE) (Kingma and Welling, 2013; Bowman et al., 2016) can be both a powerful generative model and an effective representation learning framework for natural language. In this paper, we propose the first large-scale language VAE model OPTIMUS [1]. A universal latent embedding space for sentences is first pre-trained on large text corpus, and then fine-tuned for various language generation and understanding tasks. Compared with GPT-2, OPTIMUS enables guided language generation from an abstract level using the latent vectors. Compared with BERT, OPTIMUS can generalize better on low-resource language understanding tasks due to the smooth latent space structure. Extensive experimental results on a wide range of language tasks demonstrate the effectiveness of OPTIMUS. It achieves new state-of-the-art on VAE language modeling benchmarks.

1 Introduction

Pre-trained language models (PLMs) have substantially advanced the state-of-the-art across a variety of natural language processing (NLP) tasks (Peters et al., 2018; Devlin et al., 2019; Yang et al., 2019; Radford et al., 2019; Liu et al., 2019; Keskar et al., 2019; Shoeybi et al., 2019). PLMs are often trained to predict words based on their context on massive text data, and the learned models can be fine-tuned to adapt to various downstream tasks.

PLMs can generally play two different roles: (i) a generic *encoder* such as BERT (Devlin et al., 2019) to provide contextualized representations for language understanding tasks, and (ii) a powerful *decoder* such as GPT-2 (Radford et al., 2019) to generate text sequences in an auto-regressive manner. In a bid to combine language understanding and generation tasks in one unified framework, several model variants have been proposed, including UniLM (Dong et al., 2019), BART (Lewis et al., 2019), and T5 (Raffel et al., 2019). Although significant performance improvement has been reported on a wide range of NLP tasks, these models lack of explicit modeling of structures in a latent space, rendering it difficult to control natural language generation/representation from an abstract level .

Variational Autoencoders (VAEs) (Kingma and Welling, 2013; Rezende et al., 2014) provide a tractable method to train latent-variable generative models. In NLP, latent variables may assume the role of higher-level sentence representations, which govern a lower-level word-by-word generation process, thus facilitating controlled text generation (Bowman et al., 2016; Hu et al., 2017). By representing sentences in a low-dimensional latent space, VAEs allow easy manipulation of sentences using the corresponding compact vector representations, such as the smooth feature regularization specified by prior distributions, and guided sentence generation with interpretable latent vector operators. Despite the attractive theoretical strengths, the current language VAEs are often built with small network architectures, such as two-layer LSTMs (Hochreiter and Schmidhuber, 1997). This limits the model's capacity and leads to sub-optimal performance.

In this paper, we propose OPTIMUS, the first large-scale pre-trained deep latent variable models for natural language. OPTIMUS is pre-trained using the sentence-level (variational) auto-encoder objectives on large text corpus. This leads to a universal latent space to organize sentences (hence named OPTIMUS). OPTIMUS enjoys several favorable properties: (i) It combines the strengths of VAE, BERT and GPT, and supports both natural language understanding and generation tasks. (ii) Comparing to BERT, OPTIMUS learns a more structured

[1] **O**rganizing sentences via **P**re-**T**rai**n**ed **M**odeling of a **U**niversal **S**pace

semantic space due to the use of the prior distribution in training. As a result, the language representations learned by OPTIMUS are more universal / general in that they can be more easily adapted to a new domain/task. (*iii*) Different from GPT-2, which generates human-like text but may lack effective means of controlling its high-level semantics (such as tense, topics, sentiment), OPTIMUS can be easily deployed for guided text generation.

The effectiveness of OPTIMUS has been demonstrated with extensive experiments on language modeling, dialog response generation, text style transfer and low-resource language understanding. It achieves lower perplexity than GPT-2 on standard benchmarks, produces strong performance on guided text generation, and improves BERT on feature-based language understanding tasks. The code and pre-trained models is released on Github[2] upon acceptance.

Along the way to build the first big VAE language model, there are several technical contributions/implications that are novel: (*i*) Latent vector injection: this work demonstrates two schemes to discuss how to effectively inject conditioning vectors into GPT-2 without re-training it. (*ii*) The design idea to combine BERT/GPT-2 serves as a practical recipe to inspire people to integrate and reuse existing PLMs for larger and complex models. (*iii*) Pre-training on massive datasets itself is an effective approach to reduce KL vanishing, as demonstrated by the state of-the-art performance on four VAE language modeling datasets. (*iv*) The proof of VAE objective from the lens of IB, showing that VAE is a principled approach to balance the compactness and usability of learned representations. (*v*) Improved performance on several language tasks shows the importance and necessity of pre-training a latent space.

2 Related Work

Difference with prior PLMs. Large-scale Transformer-based PLMs have recently achieved state-of-the-art performance on various natural language understanding and generation tasks (Devlin et al., 2019; Yang et al., 2019; Radford et al., 2019; Liu et al., 2019; Keskar et al., 2019). Prior to Transformer-based PLMs, non-generative methods have seen some early success in pre-training sequence models for supervised downstream tasks including standard sequence auto-encoders (Dai and Le, 2015; Li et al., 2015), skip-thought models (Kiros et al., 2015) and paragraph vector models (Le and Mikolov, 2014) *etc.* However, all of these models do not generally learn a smooth, interpretable feature space for sentence encoding, or generating novel sentences. In this work, we aim to fill the gap to learn such a universal latent space in the field of Transformer-based PLMs.

Latent variable language modeling. Language VAEs have inspired new applications in NLP, via exploiting many interesting properties of the model's latent space (Bowman et al., 2016; Kim et al., 2018b). Its modeling capacity and empirical performance is somewhat limited, partially due to the KL vanishing issue described in Section 4.3. Several attempts have been made to alleviate this issue, including different KL annealing/thresholding schemes (Bowman et al., 2016; Fu et al., 2019; Higgins et al., 2017; Li et al., 2019), decoder architectures (Yang et al., 2017; Dieng et al., 2018), auxiliary loss (Zhao et al., 2017), semi-amortized inference (Kim et al., 2018a), aggressive encoder training schedule (He et al., 2019), and flexible posterior (Fang et al., 2019). Subramanian et al. (2018) have shown some promise that general encoder can benefit language generation.

All these efforts utilize simple LSTM architectures (Hochreiter and Schmidhuber, 1997), thus with limited capacity. Our paper is the first big VAE model at the same scale of recent PLMs such as BERT and GPT-2. More importantly, we show that pre-training a meaningful latent space on a large text corpus can largely reduce the KL vanishing issue, and lead to new state-of-the-art performance.

3 Background on NLMs & GPT-2

To generate a text sequence of length T, $\boldsymbol{x} = [x_1, \cdots, x_T]$, neural language models (NLM) (Mikolov et al., 2010) generate every token x_t conditioned on the previous word tokens:

$$p(\boldsymbol{x}) = \prod_{t=1}^{T} p_{\boldsymbol{\theta}}(x_t | x_{<t}), \tag{1}$$

where $x_{<t}$ indicates all tokens before t, and $\boldsymbol{\theta}$ is the model parameter. In NLMs, each one-step-ahead conditional in (1) is modeled by an expressive family of neural networks, and is typically trained via maximum likelihood estimate (MLE). Perhaps the most well-known NLM instance is GPT-2 (Radford et al., 2019), which employs Transformers (Vaswani et al., 2017) for each conditional,

[2] https://github.com/ChunyuanLI/Optimus

and $\boldsymbol{\theta}$ is learned on a huge amount of OpenWeb text corpus. GPT-2 has shown surprisingly realistic text generation results, and low perplexity on several benchmarks.

However, the only source of variation in NLMs & GPT2 is modeled in the conditionals at every step: the text generation process only depends on previous word tokens, and there is limited capacity for the generation to be guided by the higher-level structures that are likely presented in natural language, such as tense, topics or sentiment.

4 Pre-trained Latent Space Modeling

We introduce OPTIMUS and its detailed implementation in this section. There are two stages in our framework: *pre-training* and *fine-tuning*. During *pre-training*, the model is trained on unlabeled data to construct a universal semantic space for natural language. For *fine-tuning*, the OPTIMUS model represents labeled data from the downstream tasks in the pre-trained latent space via first initialing the model with the pre-trained parameters, and fine-tuning the space by updating all/part of the parameters according to the tasks.

4.1 Pre-training Objectives

OPTIMUS organizes sentences in a universal latent (or semantic) space, via pre-training on large text corpora. Each sample in this space can be interpreted as outlines of the corresponding sentences, guiding the language generation process performed in the symbolic space (Subramanian et al., 2018). This naturally fits within the learning paradigm of latent variable models such as VAEs (Kingma and Welling, 2013; Bowman et al., 2016), where the latent representations capture the high-level semantics/patterns. It consists of two parts, generation and inference, enabling a bidirectional mapping between the latent space and symbolic space.

Generation The *generative model* (*decoder*) draws a latent vector $\boldsymbol{z}$ from the continuous latent space with prior $p(\boldsymbol{z})$, and generates the text sequence $\boldsymbol{x}$ from a conditional distribution $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$; $p(\boldsymbol{z})$ is typically assumed a multivariate Gaussian, and $\boldsymbol{\theta}$ represents the neural network parameters. The following auto-regressive decoding process is usually used:

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) = \prod_{t=1}^{T} p_{\boldsymbol{\theta}}(x_t|x_{<t}, \boldsymbol{z}). \tag{2}$$

Intuitively, VAE provides a "hierachical" generation procedure: $\boldsymbol{z} \sim p(\boldsymbol{z})$ determines the high-level semantics, followed by (2) to produce the output sentences with low-level syntactic and lexical details. This contrasts with (1) in the explicit dependency on $\boldsymbol{z}$.

Inference Similar to GPT-2, parameters $\boldsymbol{\theta}$ are typically learned by maximizing the marginal log likelihood $\log p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \log \int p(\boldsymbol{z}) p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) \mathrm{d}\boldsymbol{z}$. However, this marginal term is intractable to compute for many decoder choices. Thus, variational inference is considered, and the true posterior $p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x}) \propto p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})$ is approximated via the variational distribution $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ is (often known as the *inference model* or *encoder*), implemented via a $\boldsymbol{\phi}$-parameterized neural network. It yields the *evidence lower bound objective* (ELBO):

$$\log p_{\boldsymbol{\theta}}(\boldsymbol{x}) \geq \mathcal{L}_{\text{ELBO}} = \tag{3}$$
$$\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\big[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})\big] - \mathrm{KL}(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z}))$$

Typically, $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ is modeled as a Gaussian distribution, and the re-parametrization trick is used for efficient learning (Kingma and Welling, 2013).

A Taxonomy of Autoencoders There is an alternative interpretation of the ELBO: the VAE objective can be viewed as a regularized version of the autoencoder (AE) (Goodfellow et al., 2016). It is thus natural to extend the negative of $\mathcal{L}_{\text{ELBO}}$ in (3) by introducing a hyper-parameter β to control the strength of regularization:

$$\mathcal{L}_\beta = \mathcal{L}_E + \beta \mathcal{L}_R, \quad \text{with} \tag{4}$$
$$\mathcal{L}_E = -\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\big[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})\big] \tag{5}$$
$$\mathcal{L}_R = \mathrm{KL}(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})) \tag{6}$$

where $\mathcal{L}_E$ is the reconstruction error (or negative log-likelihood (NLL)), and $\mathcal{L}_R$ is a KL regularizer.

The cost function $\mathcal{L}_\beta$ provides a unified perspective for understanding various autoencoder variants and training methods. We consider to learn two types of latent space with the following pre-training objectives:

- **AE**. Only $\mathcal{L}_E$ is considered ($\beta = 0$), while the Gaussian sampling in $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ remains. In other words, the AE does not regularize the variational distribution toward a prior distribution, and a point-estimate is likely to be learned to represent the text sequence's latent feature. Note our reconstruction is on sentence-level, while other PLMs (Devlin et al., 2019; Yang et al., 2019) employ masked LM loss, performing token-level reconstruction.

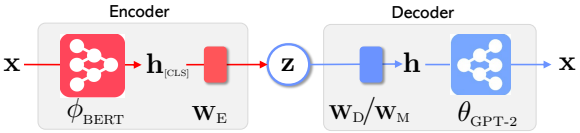


Figure 1: Illustration of OPTIMUS architecture.

- **VAE**. The full VAE objective is considered ($\beta > 0$). It tends to learn a smooth latent space due to $\mathcal{L}_R$. In practice, it is desirable to retain meaningful posteriors: the smooth structure can reduce over-fitting (Bowman et al., 2016), or generating plain dialog responses (Zhao et al., 2017).

Information Bottleneck Principle From an information theory perspective, *information bottleneck* (IB) provides a principled approach to find the trade-off between predictive power and complexity (compression) when summarizing observed data in learned representations. We show that our OPTIMUS pre-training objectives effectively practice the IB principle as follows.

The objective in (4) shows the β-VAE loss for one single sentence $\boldsymbol{x}$. The training objective over the dataset $q(\boldsymbol{x})$ can be written as:

$$\mathcal{F}_\beta = -\mathcal{F}_E + \beta \mathcal{F}_R \tag{7}$$

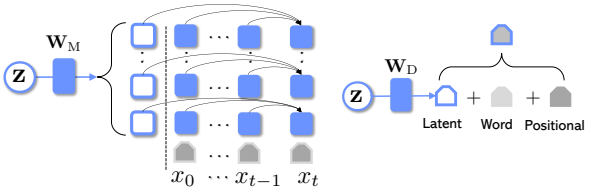
where $\mathcal{F}_E = E_{q(\boldsymbol{x}),\boldsymbol{z}\sim q(\boldsymbol{z}|\boldsymbol{x})}[\log p(\tilde{\boldsymbol{x}}|\boldsymbol{z})]$ is the aggregated reconstruction term ($\tilde{\boldsymbol{x}}$ is the reconstruction target), and $\mathcal{F}_R = \mathbb{E}_{q(\boldsymbol{x})}[\mathrm{KL}(q(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z}))]$ is the aggregated KL term. With the detailed proof shown in Section A of Appendix, we see that $\mathcal{F}_\beta$ is an upper bound of IB:

$$\mathcal{F}_\beta \geq -I_q(\boldsymbol{z}, \tilde{\boldsymbol{x}}) + \beta I_q(\boldsymbol{z}, \boldsymbol{x}) = \mathcal{L}_{\mathrm{IB}}, \tag{8}$$

where $\mathcal{L}_{\mathrm{IB}}$ is the Lagrange relaxation form of IB presented by Tishby et al. (2000), $I_q(\cdot,\cdot)$ is the mutual information (MI) measured by probability q. The goal of IB is to maximize the predictive power of $\boldsymbol{z}$ on target $\tilde{\boldsymbol{x}}$, subject to the constraint on the amount of information about original $\boldsymbol{x}$ that $\boldsymbol{z}$ carries. When $\beta = 0$, we have the AE variant of our OPTIMUS, the model fully focuses on maximizing the MI to recover sentences from the latent space. As β increases, the model gradually transits towards fitting the aggregated latent distribution $q(\boldsymbol{z}) = \int_{\boldsymbol{x}} q(\boldsymbol{z}|\boldsymbol{x})q(\boldsymbol{x})d\boldsymbol{x}$ to the given prior $p(\boldsymbol{z})$, leading the VAE variant of our OPTIMUS.

4.2 Model Architectures

The model architecture of OPTIMUS is composed of multi-layer Transformer-based encoder and decoder, based on the original implementation described in (Vaswani et al., 2017). The overall architecture is illustrated in Figure 1. To leverage the expressiveness power of existing PLMs, we initialize our encoder and decoder with weights of BERT ϕ_{BERT} and GPT-2 $\theta_{\mathrm{GPT\text{-}2}}$, respectively. This procedure is seamless, as all of these models are trained in a self-supervised/unsupervised manner.



(a) Memory (b) Embedding

Figure 2: Illustration of two schemes to inject latent vector. (a) Memory: x_t attends both $x_{<t}$ and $\boldsymbol{h}_{\mathrm{Mem}}$; (b) Embedding: latent embedding is added into old embeddings to construct new token embedding $\boldsymbol{h}'_{\mathrm{Emb}}$.

We denote the number of layers (*i.e.,* Transformer blocks) as L, the hidden size as H, and the number of self-attention heads as A. Specifically, we consider $\mathrm{BERT}_{\mathrm{BASE}}$ (L=12, H=768, A=12, Total Parameters=110M) and GPT-2 (L=12, H=768, A=12, Total Parameters=117M). We hope that our approach can provide a practical recipe to inspire future work to integrate larger pre-trained encoder and decoder for higher performance models.

Connecting BERT & GPT-2 Two technical questions remain, when pre-training OPTIMUS from BERT & GPT-2: (*i*) How to represent sentences, since the two PLMs employ different tokenization schemes? (*ii*) How to adapt a pre-trained GPT-2 to arbitrary conditional input?

Tokenization In BERT, WordPiece Embeddings (WPE) is used for tokenization (vocabulary size is 28996 for the cased version). In GPT-2, the modified Byte Pair Encoding (BPE) (Radford et al., 2019) is used for tokenization (vocabulary size is 50260). A given token is represented as $\boldsymbol{h}_{\mathrm{Emb}}$, by summing the corresponding token, position and segment embeddings [3]. For a sentence, we present it in both types of tokenization: the input of encoder is WPE, and the output of decoder is BPE to compute the reconstruction loss.

Latent Vector Injection Similar to BERT, the first token of every sentence is always a special classification token (`[CLS]`). The last-layer hidden state $\boldsymbol{h}_{\texttt{[CLS]}} \in \mathbb{R}^H$ corresponding to this to-

[3] OPTIMUS does not require segment embeddings, but we remain it due to BERT initialization.

ken is used as the sentence-level representation. It further constructs the latent representation $\boldsymbol{z} = \mathbf{W}_{\text{E}}\boldsymbol{h}_{\texttt{[CLS]}}$, where $\boldsymbol{z} \in \mathbb{R}^P$ is a P-dimensional vector and $\mathbf{W}_{\text{E}} \in \mathbb{R}^{P \times H}$ is the weight matrix. To facilitate $\boldsymbol{z}$ in GPT-2 decoding without re-training the weights, we consider two schemes, illustrated in Figure 2:

- `Memory`: $\boldsymbol{z}$ plays the role of an additional *memory* vector $\boldsymbol{h}_{\texttt{Mem}}$ for GPT2 to attend. Specifically, $\boldsymbol{h}_{\texttt{Mem}} = \mathbf{W}_{\text{M}}\boldsymbol{z}$, where $\mathbf{W}_{\text{M}} \in \mathbb{R}^{LH \times P}$ is the weight matrix. $\boldsymbol{h}_{\texttt{Mem}} \in \mathbb{R}^{LH}$ is separated into L vectors of length H, each of which is attended by GPT-2 in one layer.
- `Embedding`: $\boldsymbol{z}$ is added on the original *embedding* layer, and directly used in every decoding step. The new embedding representation is $\boldsymbol{h}'_{\texttt{Emb}} = \boldsymbol{h}_{\texttt{Emb}} + \mathbf{W}_{\text{D}}\boldsymbol{z}$, where $\mathbf{W}_{\text{D}} \in \mathbb{R}^{H \times P}$.

We study their empirical performance in Section B.1 of Appendix, and observe that `Memory` is significantly more effective than `Embedding`, and the integration of both schemes yields slightly better results. Hence, we use the integration scheme by default. In summary, the encoder parameters $\boldsymbol{\phi} = \{\boldsymbol{\phi}_{\text{BERT}}, \mathbf{W}_{\text{E}}\}$, and decoder parameters $\boldsymbol{\theta} = \{\boldsymbol{\theta}_{\text{GPT-2}}, \mathbf{W}_{\text{M}}, \mathbf{W}_{\text{D}}\}$.

4.3 Learning Procedures

We train the model parameters $\{\boldsymbol{\phi}, \boldsymbol{\theta}\}$ using two objectives: AE and VAE, discussed in Section 4.1. Pre-training AE using (5) is straightforward. However, pre-training VAE can be challenging due to the notorious *KL vanishing* issue (Bowman et al., 2016), where (i) an encoder that produces posteriors almost identical to the Gaussian prior for all sentences (rather than a more interesting posterior); and (ii) a decoder that completely ignores $\boldsymbol{z}$ in (2), and a learned model that reduces to a simpler NLM.

To reduce this issue, we follow the intuition that if the encoder is providing useful information from the beginning of decoder training, the decoder is more likely to make use of $\boldsymbol{z}$ (Fu et al., 2019; He et al., 2019). Specifically, we use the cyclical schedule to anneal β for 10 periods (Fu et al., 2019). Within one period, there are three consecutive stages: Training AE ($\beta = 0$) for 0.5 proportion, annealing β from 0 to 1 for 0.25 proportion, and fixing $\beta = 1$ for 0.25 proportion. When $\beta > 0$, we use the KL thresholding scheme (Li et al., 2019; Kingma et al., 2016), and replace the KL term $\mathcal{L}_R$ in (6) with a hinge loss term that maxes each component of the original KL with a constant λ:

$$\mathcal{L}'_R = \sum_i \max[\lambda, \text{KL}(q_{\boldsymbol{\phi}}(z_i|\boldsymbol{x})||p(z_i))] \quad (9)$$

Here, z_i denotes the ith dimension of $\boldsymbol{z}$. Using the thresholding objective causes learning to give up driving down KL for dimensions of $\boldsymbol{z}$ that are already beneath the target compression rate.

Pre-training data The pre-training procedure largely follows the existing literature on language model pre-training. We use English Wikipedia to pre-train our AE and VAE objectives. As our main interest is to model sentences (rather than text sequences of a fixed length), we pre-process Wikipedia with maximum sentences length 64. It leads to 1990K sentences, which accounts 96.45% Wikipedia sentences used in BERT. More data preprocessing details are in Section B.2 of Appendix.

5 Fine-tuning OPTIMUS

5.1 Language Modeling

Fine-tuning LM on new datasets is straightforward. We load the pre-trained OPTIMUS, and update the model with one additional β scheduling cycle for one epoch. The semantic latent vectors are first pre-trained off-the-shelf, and then easily leveraged to train the decoder on downstream datasets. From this perspective, our pre-training can be viewed as an effective approach to reduce KL vanishing.

5.2 Guided Language Generation

Different from the traditional NLMs, VAEs learns bidirectional mappings between the latent and symbolic space. It enables high-level sentence editing as arithmetic latent vector operations, and thus allows guided language generation. We consider more sophisticated latent space manipulation in two applications, with details in Appendix.

(Stylized) Response Generation The open-domain dialog response generation task in two settings: (i) generating responses $\boldsymbol{x}$ given a dialog history $\boldsymbol{c}$; (ii) Similar to (i) , with an additional requirement that the generated responses are in the style specified by the corpus $\boldsymbol{b}$. Following (Gao et al., 2019a,b), we embed the history, response and style-reference in a joint latent space as $\boldsymbol{z}_{\text{S2S}}$, $\boldsymbol{z}_{\text{AE}}$ and $\boldsymbol{z}_{\text{Style}}$, respectively. A fusion regularization is used to match the (stylized) responses to the context in the latent space.

	Dataset	PTB			YELP			YAHOO			SNLI		
		LM	Repr.		LM	Repr.		LM	Repr.		LM	Repr.	
	Method	PPL ↓	MI ↑	AU ↑	PPL ↓	MI ↑	AU ↑	PPL ↓	MI ↑	AU ↑	PPL ↓	MI ↑	AU ↑
OPTIMUS	$\lambda=0.05$	**23.58**	3.78	32	**21.99**	2.54	32	22.34	5.34	32	**13.47**	3.49	32
	$\lambda=0.10$	23.66	4.29	32	21.99	2.87	32	22.57	5.35	32	13.48	4.65	32
	$\lambda=0.25$	24.34	5.98	32	22.20	5.31	32	22.43	6.01	32	14.08	7.22	32
	$\lambda=0.50$	26.69	7.64	32	22.79	7.67	32	23.11	8.85	32	16.67	8.89	32
	$\lambda=1.00$	35.53	**8.18**	32	24.59	**9.13**	32	24.92	**9.18**	32	29.63	**9.20**	32
Small VAE	M. A.	101.40	0.00	0	40.39	0.13	1	61.21	0.00	0	21.50	1.45	2
	C. A.	108.81	1.27	5				66.93	2.77	4	23.67	3.60	5
	SA-VAE					1.70	8	60.40	2.70	10			
	Aggressive	99.83	0.83	4	39.84	2.16	12	59.77	2.90	19	21.16	1.38	5
	AE-BP	96.86	5.31	32	47.97	7.89	32	59.28	8.08	32	21.64	7.71	32
	GPT-2	24.23	-	-	23.40	-	-	22.00	-	-	19.68	-	-
	LSTM-LM	100.47	-	-	42.60	-	-	60.75	-	-	21.44	-	-
	LSTM-AE	-	8.22	32	-	9.24	32	-	9.26	32	-	9.18	32

Table 1: Comparison on language modeling tasks on four datasets. Best values are in blue. $\lambda = 0.50$ is a good trade-off to achieve the best values on all metrics compared with small VAEs. "-" indicates the models are improper to report these values; Empty cells indicate the results were not reported in the literature.

Label-Conditional Text Generation We fine-tune using the VAE objective on a new labeled dataset, then freeze OPTIMUS weights. A conditional GAN (Mirza and Osindero, 2014) is trained on the fixed latent space. The generation process is to first produce a latent vector $\boldsymbol{z}_y$ based on a given label y using conditional GAN, then generate sentences conditioned on $\boldsymbol{z}_y$ using the decoder.

5.3 Low-resource Language Understanding

Due to the regularization term $\mathcal{L}_R$, OPTIMUS can organize sentences in the way defined by the prior distribution in the latent space. In the case of VAEs, a smooth feature space is learned, which is specifically beneficial for better generalization when the number of task-specific labeled data is low. Following BERT, the `[CLS]` representation $\boldsymbol{h}_{\texttt{[CLS]}}$ is fed into an linear layer $\mathbf{W}_{\text{C}} \in \mathbb{R}^{K \times H}$ for classification, where K is the number of classes. The classification loss is $-\log(\text{softmax}(\boldsymbol{h}_{\texttt{[CLS]}})\mathbf{W}_{\text{C}}^{\top})$.

6 Experimental Results

6.1 Language Modeling

We consider four datasets: the Penn Treebank (`PTB`) (Marcus et al., 1993), `SNLI` (Bowman et al., 2015), `Yahoo`, and `Yelp` corpora (Yang et al., 2017; He et al., 2019).

Metrics There are two types of metrics to evaluate language VAEs. (i) Generation capability: we use *perplexity* (PPL). Note that NLM and GPT-2 has exactly PPL, while VAEs does not. Following (He et al., 2019), we use the importance weighted bound in (Burda et al., 2015) to approximate $\log p(\boldsymbol{x})$, and report PPL. (ii) Representation learning capability: Active units (AU) of $\boldsymbol{z}$ and its Mutual Information (MI) with $\boldsymbol{x}$. We report the full results with ELBO, KL and Reconstruction in Appendix, but note that ELBO does not necessarily yield better language modeling performance.

Baseline Methods (i) *GPT-2*. A large-scale LM trained on OpoenWebText (Radford et al., 2019). We load the pre-trained GPT-2 weights, and refine the model for 1 epoch on the new datasets. (ii) *Annealing*. β is gradually annealed from 0 to 1. This annealing procedure can be used once (M.A.) (Bowman et al., 2016) or multiple times (C.A.) (Fu et al., 2019). (iii) *Aggressive Training* (He et al., 2019). Training the encoder multiple times per decoder update. (iv) *AE-FB* (Li et al., 2019). Training AE, and then VAE using the KL thresholding in (9), the results on $\lambda = 5$ are reported as a good trade-off..

The results are shown in Table 1. Various λ values are used, we observe a trade-off between language modeling and representation learning, controlled by λ. Compared with existing VAE methods, OPTIMUS achieve significantly lower perplexity, and higher MI/AU. This indicates that our pre-training method is an effective approach to reduce KL vanishing issue and training VAEs, especially given the fact that we only fine-tune on these datasets for one epoch. OPTIMUS achieves lower perplexity compared with GPT-2 on three out of four datasets. Intuitively, this is because the model can leverage the prior language knowl-

Source $\boldsymbol{x}_A$	**Target** $\boldsymbol{x}_B$
a girl makes a silly face	two soccer players are playing soccer
Input $\boldsymbol{x}_C$	**Output** $\boldsymbol{x}_D$
• a girl poses for a picture	• two soccer players are at a soccer game.
• a girl in a blue shirt is taking pictures of a microscope	• two football players in blue uniforms are at a field hockey game
• a woman with a red scarf looks at the stars	• two men in white uniforms are field hockey players
• a boy is taking a bath	• two baseball players are at the baseball diamond
• a little boy is eating a bowl of soup	• two men are in baseball practice

Table 2: Sentence transfer via arithmetic operation in the latent space. The output sentences are in blue.

0.0	children are looking for the water to be clear.
0.1	children are looking for the water.
0.2	children are looking at the water.
0.3	the children are looking at a large group of people.
0.4	the children are watching a group of people.
0.5	the people are watching a group of ducks.
0.6	the people are playing soccer in the field.
0.7	there are people playing a sport.
0.8	there are people playing a soccer game.
0.9	there are two people playing soccer.
1.0	there are two people playing soccer.

Table 3: Interpolating latent space. Each row shows τ, and the generated sentence (in blue) conditioned on $\boldsymbol{z}_\tau$.

edge encoded in $\boldsymbol{z}$. This gap is larger, when the sentences in the dataset exhibit common regularities, such as SNLI, where the prior plays a more important/effective role in this scenario. Though the form of our model is simple, OPTIMUS shows stronger empirical performance than models that are particularly designed for long-text, such as hierarchical VAE (hVAE) in (Shen et al., 2019). For example, the KL and PPL of OPTIMUS (15.09 and 22.79) are much better than hVAE (6.8 and 45.8) on Yelp dataset. This verifies the importance of pre-training a latent space. The full experimental results are shown in Table 8, 9, 10 and 11 of Appendix.

6.2 Guided Language Generation

Arithmetic operation The universal latent space learned by OPTIMUS supports arithmetic operations. Given source sentence $\boldsymbol{x}_A$ and target $\boldsymbol{x}_B$, the goal is to re-write the input sentence $\boldsymbol{x}_C$ as output $\boldsymbol{x}_D$ in analogy to the transition from $\boldsymbol{x}_A$ to $\boldsymbol{x}_B$. We first encode $\boldsymbol{x}_{A,B,C}$ into the latent vectors $\boldsymbol{z}_{A,B,C}$, respectively, then apply the arithmetic operator $\boldsymbol{z}_D = \boldsymbol{z}_B - \boldsymbol{z}_A + \boldsymbol{z}_C$, and generate $\boldsymbol{x}_D$ conditioned on $\boldsymbol{z}_D$. One example is shown in Table 2. Interestingly, we observe consistent style transfer from $\boldsymbol{x}_C$ to $\boldsymbol{x}_D$, to analogize the relation from $\boldsymbol{x}_A$ to $\boldsymbol{x}_B$. For example, the subject is revised from singular to plural forms, the topic changes from daily-life to sport. In another word, OPTIMUS supports sentence arithmetic operator $\boldsymbol{x}_D \approx \boldsymbol{x}_B - \boldsymbol{x}_A + \boldsymbol{x}_C$ at the semantic level. More examples are shown in Appendix.

Latent space interpolation One favorable property of VAEs is to provide a smooth space that captures sentence semantics. We demonstrate linear interpolating between latent vectors. We take two sentences $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, and use their posterior mean as the latent features $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$, respectively. We interpolate a path $\boldsymbol{z}_\tau = \boldsymbol{z}_1 \cdot (1-\tau) + \boldsymbol{z}_2 \cdot \tau$ with τ increased from 0 to 1 by a step size of 0.1. Table 3 shows generated sentences using greedy decoding conditioned on $\boldsymbol{z}_\tau$. The interpolated sentences exhibit smooth semantic evolution. More interpolation examples are shown in Appendix. Note that we have observed smooth & meaningful interpolation results for almost arbitrary input sentences pairs. This demonstrates the promise that OPTIMUS learns a universal latent space.

Arithmetic operation and interpolation are two simple schemes to manipulate pre-trained latent spaces. They showcases that OPTIMUS enables new ways that one can play with language generation using pre-trained models, compared with GPT-2 that can only fulfill text sequences with given prompts. We demonstrate more sophisticated ways to manipulate pre-trained latent spaces in three real applications as follows.

Dialog response generation We consider Dailydialog (Li et al., 2017c) used in (Gu et al., 2019), which has 13,118 daily conversations. Each utterance is processed as the response of previous 10 context utterances from both speakers. The baseline methods are described in Appendix. We measure the performance using Bleu (Chen and Cherry, 2014), and compute the precision, recall and F1 in Table 4. OPTIMUS shows higher Bleu scores than all existing baselines.

Metrics	Seq2Seq	CVAE	WAE	iVAE$_{MI}$	OPTIMUS
Recall↑	0.232	0.265	0.289	0.355	**0.362**
Precision↑	0.232	0.222	0.266	0.239	**0.313**
F1↑	0.232	0.242	0.277	0.285	**0.336**

Table 4: Dialog response generation on `DailyDialog` dataset. All numbers are from (Gu et al., 2019) except that iVAE$_{MI}$ is from (Fang et al., 2019).

Methods	Recall↑	Precision↑	F1↑	Neural↑	N-gram↑
StyleFusion	0.374	0.242	0.294	0.1050	0.1495
OPTIMUS	**0.385**	**0.268**	**0.316**	**0.1191**	**0.1645**

Table 5: Stylized response generation.

Metrics	Control-Gen	ARAE	NN-Outlines	OPTIMUS
Accuracy↑	0.878	0.967	0.553	**0.998**
Bleu ↑	0.389	0.201	0.198	**0.398**
G-score↑	0.584	0.442	0.331	**0.630**
Self-Bleu↓	0.412	0.258	0.347	**0.243**

Table 6: Label-conditional text generation on `Yelp`.

Stylized response generation Following StyleFusion (Gao et al., 2019b), we consider generating responses for `Dailydialog` in the style of Holmes. The comparison is shown in Table 5. In addition to Bleu, we use neural and N-gram classifier scores to evaluate the accuracy of the generated responses that belong to the desired style. OPTIMUS achieves better performance on all metrics.

Label-conditional text generation We consider the short `Yelp` dataset collected in (Shen et al., 2017). It contains 444K training sentences, and we use separated datasets of 10K sentences for validation/testing, respectively. The goal is to generate text reviews given the positive/negative sentiment. The baselines are described in Appendix. G-score computes the geometric mean of Accuracy and Bleu, measuring the comprehensive quality of both content and style. Self-Bleu measures the diversity of the generated sentences. The results are shown in Table 6, OPTIMUS achieves the best performance on all metrics. The conditional generated sentences are shown in Appendix.

6.3 Low-resource Language Understanding

Sentiment classification on Yelp dataset. A linear classifier is added on the feature of `[CLS]` token. A various number of samples are randomly chosen for training, ranging from 1 to 10K per class. 10 runs are used when the number of available training samples are small. 100 training epochs are used in each setting. Two schemes are used to leverage pre-trained models: (i) *Fine-tuning*, where both the pre-trained model and the linear classifier are updated; (ii) *Feature-based*, where pre-trained model weights are frozen to provide embeddings for the update of the linear classifier.

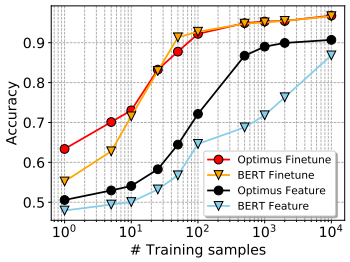
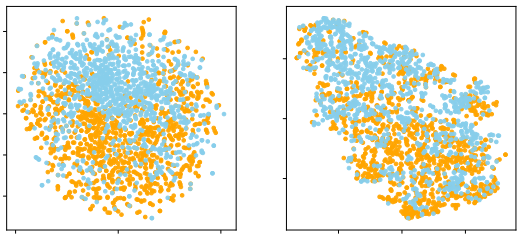


Figure 3: Testing accuracy with a varying number of labeled training samples per class on the `Yelp` dataset.



(a) OPTIMUS (b) BERT

Figure 4: Comparison of tSNE visualization for the self-supervised feature learning results. The colors indicate different labels.

The results are shown in Figure 3. When pre-trained models are used to provide sentence embeddings, the proposed OPTIMUS consistently outperforms BERT. It demonstrates that the latent structure learned by OPTIMUS is more separated, and helps generalize better. When the entire network is fine-tuned, OPTIMUS can adapt faster than BERT, when the available number of training samples is small. The two methods perform quite similarly when more training data is provided. This is because the pre-trained backbone model size is much larger than the linear classifier, where the performance is largely dominated by the backbone networks in training.

Visualization of the latent space. We use tSNE (Maaten and Hinton, 2008) to visualize the learned feature on a 2D map. The validation set of Yelp is used to extract the latent features. Compared with BERT, OPTIMUS learns a smoother space and more structured latent patterns, which explains why OPTIMUS can yield better classification performance and faster adaptation.

GLUE. We further consider the GLUE benchmark (Wang et al., 2019), which consists of nine datasets for general language understanding. Following the finetuning schedule in (Devlin et al., 2019), we use learning rate $[2, 3, 4, 5] \times 10^{-5}$ and

System / Dataset size		MNLI 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	WNLI 634	**Average**
Feature-based	BERT	0.414	0.146	0.673	0.731	0.187	0.690	0.812	0.549	0.577	0.531± 0.011
	OPTIMUS	**0.468**	**0.662**	**0.720**	**0.789**	0.144	**0.719**	**0.816**	**0.585**	0.563	**0.607**± 0.013

Table 7: Comparison on the validation set of GLUE. F1 scores are reported for QQP and MRPC, Spearman correlations are reported for STS-B, and accuracy scores are reported for the other tasks.

train the model for 3 epochs. We select the best performance among different runs. We show the results on the validation set in Table 7. With the feature-based scheme, OPTIMUS yields higher performance than BERT, especially on the large datasets such as MNLI, QQP and QNLI. When the full models are fine-tuned, the two methods perform quite similarly (shown in Table 21 of Appendix).

In summary, the scenarios that OPTIMUS fit the low-resource settings are two-fold: (1) The required computing resource is low: the âĂIJfeature-basedâĂİ approach only updates the classifier, whose computing requirement is much lower than full-model fine-tuning; (2) The number of required labelled data is low: when labelled data is rare, OPTIMUS adapts better. The results confirm that OPTIMUS can maintain and exploit the structures learned in pre-training, and presents a more general representation that can be adapted to new tasks more easily than BERT – feature-based adaption is much faster and easier to perform than fine-tuning.

7 Discussion

We present OPTIMUS, a large-scale pre-trained deep latent variable model for natural language. It introduces a smooth and universal latent space, by combining the advantages of VAEs, BERT and GPT-2 in one model. Experimental results on a wide range of tasks and datasets have demonstrated the strong performance of OPTIMUS, including new state-of-the-art for language VAEs.

While deep generative models (DGMs) such as VAEs are theoretically attractive for NLP community due to its principle nature, it is now rarely used by practitioners in the modern pre-trained language modeling era where BERT/GPT dominate with strong empirical performance. That's why this paper makes a timely contribution to making DGMs practical for NLP. We hope that this paper will help renew interest in DGMs for this purpose. Hence, we deliberately keep a simple model, believing that the first pre-trained big VAE model itself and its implications are novel: it helps the community to recognize the importance of DGMs in the pre-training era, and revisit DGMs to make it more practical. Indeed, Optimus is uniquely positioned to learn a smooth latent space to organize sentences, which can enable guided language generation compared with GPT-2, and yield better generalization in low-resource language understanding tasks than BERT.

References

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. *CONLL*.

Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. 2015. Importance weighted autoencoders. *ICLR*.

Boxing Chen and Colin Cherry. 2014. A systematic comparison of smoothing techniques for sentence-level Bleu. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*.

Ricky TQ Chen, Xuechen Li, Roger Grosse, and David Duvenaud. 2018. Isolating sources of disentanglement in VAEs. *NIPS*.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *NIPS*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. *NAACL*.

Adji B Dieng, Yoon Kim, Alexander M Rush, and David M Blei. 2018. Avoiding latent variable collapse with generative skip models. *AISTATS*.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. *NeurIPS*.

Le Fang, Chunyuan Li, Jianfeng Gao, Wen Dong, and Changyou Chen. 2019. Implicit deep latent variable models for text generation. *EMNLP*.

Hao Fu, Chunyuan Li, Xiaodong Liu, Jianfeng Gao, Asli Celikyilmaz, Lawrence Carin, et al. 2019. Cyclical annealing schedule: A simple approach to mitigating KL vanishing. *NAACL*.

Xiang Gao, Sungjin Lee, Yizhe Zhang, Chris Brockett, Michel Galley, Jianfeng Gao, and Bill Dolan. 2019a. Jointly optimizing diversity and relevance in neural response generation. *NAACL*.

Xiang Gao, Yizhe Zhang, Sungjin Lee, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2019b. Structuring latent spaces for stylized response generation. *EMNLP*.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep learning*. MIT press Cambridge.

Xiaodong Gu, Kyunghyun Cho, Jungwoo Ha, and Sunghun Kim. 2019. DialogWAE: Multimodal response generation with conditional wasserstein autoencoder. *ICLR*.

Junxian He, Daniel Spokoyny, Graham Neubig, and Taylor Berg-Kirkpatrick. 2019. Lagging inference networks and posterior collapse in variational autoencoders. *ICLR*.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. 2017. beta-vae: Learning basic visual concepts with a constrained variational framework. *ICLR*.

Sepp Hochreiter and Jurgen Schmidhuber. 1997. Long short-term memory. *Neural computation*.

Matthew D Hoffman and Matthew J Johnson. 2016. Elbo surgery: yet another way to carve up the variational evidence lower bound. In *Workshop in Advances in Approximate Bayesian Inference, NIPS*.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. *ICML*.

Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, and Richard Socher. 2019. Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*.

Yoon Kim, Sam Wiseman, Andrew C Miller, David Sontag, and Alexander M Rush. 2018a. Semi-amortized variational autoencoders. *ICML*.

Yoon Kim, Sam Wiseman, and Alexander M Rush. 2018b. A tutorial on deep latent variable models of natural language. *arXiv preprint arXiv:1812.06834*.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *ICLR*.

Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. 2016. Improved variational inference with inverse autoregressive flow. In *NIPS*.

Ryan Kiros, Yukun Zhu, Russ R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *NIPS*.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *ICML*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Bohan Li, Junxian He, Graham Neubig, Taylor Berg-Kirkpatrick, and Yiming Yang. 2019. A surprisingly effective fix for deep latent variable modeling of text. *EMNLP*.

Chunyuan Li, Hao Liu, Changyou Chen, Yuchen Pu, Liqun Chen, Ricardo Henao, and Lawrence Carin. 2017a. ALICE: Towards understanding adversarial learning for joint distribution matching. In *NIPS*.

Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. 2015. A hierarchical neural autoencoder for paragraphs and documents. *ACL*.

Jiwei Li, Will Monroe, Tianlin Shi, Sébastien Jean, Alan Ritter, and Dan Jurafsky. 2017b. Adversarial learning for neural dialogue generation. *arXiv preprint arXiv:1701.06547*.

Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. 2017c. Dailydialog: A manually labelled multi-turn dialogue dataset. *arXiv preprint arXiv:1710.03957*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *JMLR*.

Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. 2016. Adversarial autoencoders. *ICLR workshop*.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *InterSpeech*.

Mehdi Mirza and Simon Osindero. 2014. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*.

Yookoon Park, Jaemin Cho, and Gunhee Kim. 2018. A hierarchical latent structure for variational conversation modeling. *arXiv preprint arXiv:1804.03424*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. *ICML*.

Iulian Vlad Serban, Alessandro S., Yoshua Bengio, Aaron C Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*.

Iulian Vlad Serban, A. Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *AAAI*.

Dinghan Shen, Asli Celikyilmaz, Yizhe Zhang, Liqun Chen, Xin Wang, Jianfeng Gao, and Lawrence Carin. 2019. Towards generating long and coherent text with multi-level latent variable models. *ACL*.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *NIPS*.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. Megatron-LM: Training multi-billion parameter language models using gpu model parallelism. *arXiv preprint arXiv:1909.08053*.

Sandeep Subramanian, Sai Rajeswar Mudumba, Alessandro Sordoni, Adam Trischler, Aaron C Courville, and Chris Pal. 2018. Towards text generation with adversarially learned neural outlines. In *NeurIPS*.

Naftali Tishby, Fernando C Pereira, and William Bialek. 2000. The information bottleneck method. *arXiv preprint physics/0004057*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. *ICLR*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. XLNet: Generalized autoregressive pretraining for language understanding. *NeurIPS*.

Zichao Yang, Zhiting Hu, Ruslan Salakhutdinov, and Taylor Berg-Kirkpatrick. 2017. Improved variational autoencoders for text modeling using dilated convolutions. *ICML*.

Jake Zhao, Yoon Kim, Kelly Zhang, Alexander M Rush, and Yann LeCun. 2018. Adversarially regularized autoencoders. *ICML*.

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. *ACL*.

Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. 2018. Texygen: A benchmarking platform for text generation models. In *ACM SIGIR*.

A Information Bottleneck and VAEs

Definition of IB Tishby et al. (2000) presented the Information Bottleneck (IB) method via solving the Lagrange relaxation of the optimization problem:

$$\min \mathcal{L}_{\text{IB}} = -I(\boldsymbol{z}; \tilde{\boldsymbol{x}}) + \beta I(\boldsymbol{z}; \boldsymbol{x}) \tag{10}$$

where $\boldsymbol{z}$ is the representation of $\boldsymbol{x}$, and β is a positive parameter that controls the trade-off between the compression of input $\boldsymbol{x}$ and preserved information about target $\tilde{\boldsymbol{x}}$.

In the following, we first show that the KL and reconstruction terms of VAE are the bounds of MI, respectively. Further, we put the bounds together, and show that VAE objective can optimize BI.

KL upper bounds MI Following (Makhzani et al., 2016), we refer to $q(\boldsymbol{z}) = \int_{\boldsymbol{x}} q(\boldsymbol{z}|\boldsymbol{x})q(\boldsymbol{x})d\boldsymbol{x}$ as the aggregated posterior. This marginal distribution captures the aggregated $\boldsymbol{z}$ over the entire dataset. The KL term (6) in can be decomposed into two refined terms (Chen et al., 2018; Hoffman and Johnson, 2016):

$$\begin{aligned}\mathcal{F}_R &= \mathbb{E}_{q(\boldsymbol{x})}[\text{KL}(q(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z}))] \\ &= \underbrace{I_q(\boldsymbol{z}, \boldsymbol{x})}_{\mathcal{F}_1\text{: Mutual Info.}} + \underbrace{\text{KL}(q(\boldsymbol{z})||p(\boldsymbol{z}))}_{\mathcal{F}_2\text{: Marginal KL}} \\ &\geq I_q(\boldsymbol{z}, \boldsymbol{x})\end{aligned} \tag{11}$$

where $\mathcal{F}_1$ is the mutual information (MI) measured by q. Higher MI can lead to a higher correlation between the latent variable and data variable, and encourages a reduction in the degree of KL vanishing. The marginal KL is represented by $\mathcal{F}_2$, and it measures the fitness of the aggregated posterior to the prior distribution.

Reconstruction lower bounds MI The reconstruction term in (5) provides a lower bound for MI measured by q, based on Corollary 3 in (Li et al., 2017a):

$$\begin{aligned}\mathcal{F}_E &= E_{q(\boldsymbol{x}), \boldsymbol{z}\sim q(\boldsymbol{z}|\boldsymbol{x})}(\log p(\tilde{\boldsymbol{x}}|\boldsymbol{z}))] + H_q(\tilde{\boldsymbol{x}}) \\ &\leq I_q(\boldsymbol{z}, \tilde{\boldsymbol{x}})\end{aligned} \tag{12}$$

where $\tilde{\boldsymbol{x}}$ is the reconstruction target in our auto-encoder setting, and $H(\tilde{\boldsymbol{x}})$ is a constant.

VAE recovers BI When scheduled with β, the training objective over the dataset can be written as:

$$\mathcal{F}_\beta = -\mathcal{F}_E + \beta\mathcal{F}_R \tag{13}$$

$$\geq -I_q(\boldsymbol{z}, \tilde{\boldsymbol{x}}) + \beta I_q(\boldsymbol{z}, \boldsymbol{x}) \tag{14}$$

This recovers IB principle in (10). When $\beta = 0$, we have the AE variant of our OPTIMUS, the model fully focuses on maximizing the MI to recover sentence from the latent space. As β increases, the model gradually transits towards fitting the aggregated latent codes to the given prior, leading the VAE variant of our OPTIMUS.

B Pre-training Details

B.1 Latent Vector Injection Schemes

We compare three different schemes to inject latent vector into GPT2 in Figure 5:

- **Mem**. Latent vector $\boldsymbol{z}$ is used as additional *memory* token for GPT2 to attend.
- **Emb**. Latent vector $\boldsymbol{z}$ is used as additional *embedding* to add into other embeddings.
- **Mem+Emb**. The integration of the above two schemes.

On both Yelp and PTB datasets, 5 training epochs are considered. Yelp generally has longer sentences than PTB. The encoder is initialized with BERT, and decoder is initialized with GPT-2. Lower reconstruction error per word indicates a more effective approach to pass the information flow from encoder to decoder. We see that it is significantly more efficient to use $\boldsymbol{z}$ as a memory vector for GPT-2 to attend, than as the additional embedding. The combined scheme yields slightly better performance in the late stage of training. In the paper, we use the combined scheme in default.

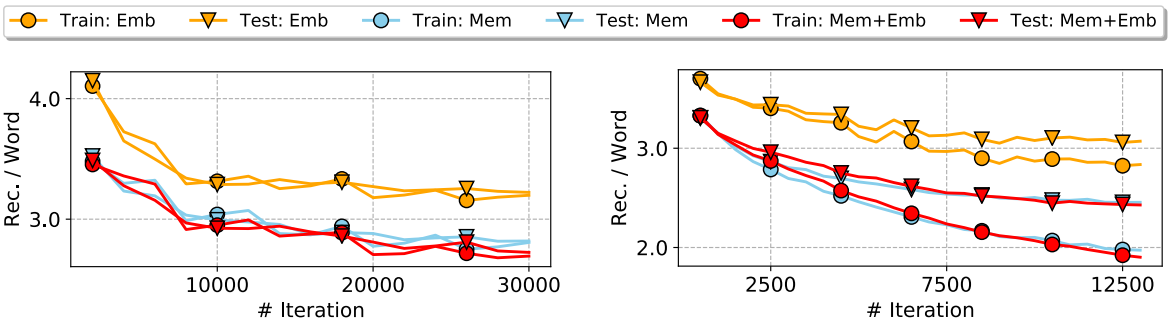
B.2 Wikipedia Dataset

We illustrate the statistics of Wikipedia dataset in Figure 6. Since we focus on modeling natural sentences (rather than text sequences of a fixed length as in GPT-2 (Radford et al., 2019)) in a latent space, we pre-process Wikipedia into a set of natural sentences, with maximum sequence length as 64. This leads to 1990K sentences, which is 96.45% of entire Wikipedia dataset.

C Experiment Details

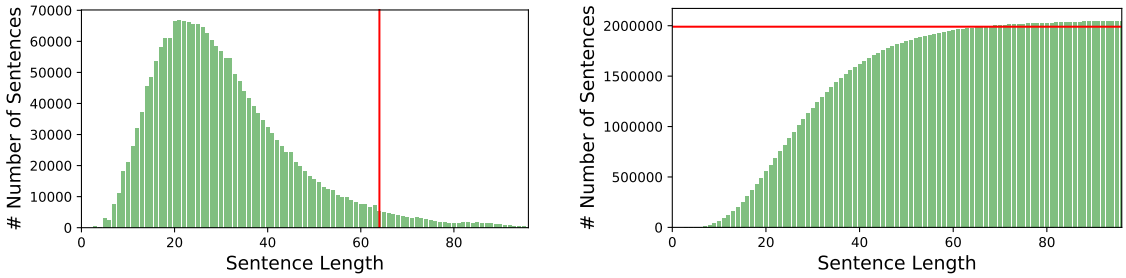
C.1 Language Modeling

In addition to generating high-quality sentences as in the traditional language models that only, VAEs also aim to learn a good posterior distribution in the latent space. The language modeling performance is evaluated with ELBO, perplexity (PPL) or importance weighted perplexity (He et al., 2019),



(a) Yelp (b) PTB

Figure 5: Illustration of three different schemes to inject latent vector into GPT-2 for guided language generation: (a) Yelp and (b) PTB. The learning curves for reconstruction error per word is considered. `Emb` indicates latent vector is used as additional embedding to add into other embeddings, and `Mem` indicates latent vector is used as additional memory token for GPT2 to attend. `Mem+Emb` indicates the integration of two schemes.



(a) Frequency distribution (b) Cumulative frequency distribution

Figure 6: Illustration of sentence distribution in Wikipedia dataset: (a) Frequency distribution and (b) Cumulative Frequency distribution. We choose maximum length as 64 to construct the pre-training dataset. It leads to 1990K sentences, which is 96.45% of entire Wikipedia dataset.

which provides a tighter bound to $\log p(\boldsymbol{x})$. Higher ELBO and lower PPL indicate the model fits the observed sentences better.

- **ELBO**: The sum of KL divergence and reconstruction loss.
- **Perplexity.** PPL $= p(x_1, \cdots, x_N)^{-1/N}$, where N is the number of words. For latent variable models, we use a lower bound on the marginal log-likelihood $\log p(\boldsymbol{x})$, as follows from Jensen's Inequality and the fact that the average importance weights are an unbiased estimator of $p(\boldsymbol{x})$:

$$\mathcal{L}_k = \mathbb{E}\big[\log \frac{1}{k}\sum_{i=1}^{k} w_i\big]$$
$$\leq \log\big[\mathbb{E}\frac{1}{k}\sum_{i=1}^{k} w_i\big] = \log p(\boldsymbol{x}). \quad (15)$$

where $w_i = p(\boldsymbol{x}, \boldsymbol{z}_i)/q(\boldsymbol{z}_i|\boldsymbol{x})$.

More importantly, we are interested in the learned $\boldsymbol{z}$, which is evaluated using the following three metrics:

- **AU**: The total number of active units in $\boldsymbol{z}$, defined as $A_z = \mathrm{Cov}_{\boldsymbol{x}}(\mathbb{E}_{z\sim q(z|\boldsymbol{x})}[z]) > 0.01$ (Burda et al., 2015);
- **MI**: The mutual information $I(\boldsymbol{x}, \boldsymbol{z})$;
- **KL**: The posterior-prior KL divergence

The full experimental results on shown in Table 8, 9, 10 and 11.

C.2 Dialog response generation

Dialog response generation: SpaceFusion We interpolate samples $\boldsymbol{z}_\tau$ between the context and response as $\boldsymbol{z}_\tau = \tau\boldsymbol{z}_{\text{S2S}} + (1-\tau)\boldsymbol{z}_{\text{AE}}$, where $\tau \sim \text{Uniform}(0, 1)$. We fix the first 11 layers of encoder, and fine-tune from last layer to $\boldsymbol{z}$: $\{\phi_{\text{AE}}, \phi_{\text{E}}\}$. An additional network path $\{\phi_{\text{S2S}}, \phi'_{\text{E}}\}$ is introduced from the 11th layer of encoder to $\boldsymbol{z}$ to represent context. The fine-tuning objective is:

$$\min_{\{\phi_{\text{S2S}}, \phi_{\text{AE}}, \phi_{\text{E}}, \phi'_{\text{E}}, \boldsymbol{\theta}\}} \mathcal{L}_{\text{dialog}} = \mathcal{L}_{\boldsymbol{x}} + \mathcal{L}_{\text{fusion}}$$

Metric	LM	Representation		Learning Objective		
Method	PPL ↓	MI ↑	AU ↑	-ELBO ↓	KL ↑	Rec ↓
Ours($\lambda=0.05$)	23.58	3.78	32	91.31	4.88	86.43
Ours($\lambda=0.1$)	23.66	4.29	32	91.60	5.82	85.78
Ours($\lambda=0.25$)	24.24	5.98	32	93.18	9.42	83.75
Ours($\lambda=0.5$)	26.69	7.64	32	96.82	15.72	81.09
Ours($\lambda=1.0$)	35.53	8.17	32	77.65	28.50	77.65
GPT-2	24.23					
LSTM-LM	100.47			101.04		
LSTM-AE		8.22	32			70.36
M. Annealing	101.40	0.0	0	101.28	0.0	101.28
C. Annealing	108.81	1.27	5	102.81	1.37	101.85
Aggressive	99.83	0.83	4	101.19	0.93	100.26
AE-BP ($\lambda=5$)	96.86	5.31	32	102.41	6.54	95.87

Table 8: Comparison on PTB dataset.

Metric	LM	Representation		Learning Objective		
Method	PPL ↓	MI ↑	AU ↑	-ELBO ↓	KL ↑	Rec ↓
Ours($\lambda=0.01$)	21.99	2.54	32	337.41	3.09	334.31
Ours($\lambda=0.05$)	21.99	2.87	32	337.61	3.73	333.87
Ours($\lambda=0.25$)	22.20	5.31	32	340.03	8.70	331.33
Ours($\lambda=0.5$)	22.79	7.67	32	344.10	15.09	329.01
Ours($\lambda=1.0$)	24.59	9.13	32	353.67	27.89	325.77
GPT-2	23.40					
LSTM-LM				358.10		
LSTM-AE		9.26	32			278.76
SA-VAE		1.7	8	355.90	2.80	353.10
M. Annealing	40.39	0.13	1	357.76	0.14	357.62
C. Annealing						
Aggressive		2.4	7	328.40	3.4	322.70
AE-BP ($\lambda=5$)						

Table 9: Comparison on Yelp dataset. For LSTM-LM and GPT-2, we report the exact negative log likelihood.

where $\mathcal{L}_{\text{fusion}}$ is the same with fusion term in (Gao et al., 2019a), and $\mathcal{L}_{\boldsymbol{x}} = -[\log p(\boldsymbol{x}|\boldsymbol{z}_{\text{S2S}}) + \log p(\boldsymbol{x}|\boldsymbol{z}_{\text{AE}}) + \log p(\boldsymbol{x}|\boldsymbol{z}_{\tau})]$.

We benchmark representative baselines and state-of-the-art approaches, including: (i) Seq2Seq: a generalized sequence-to-sequence model with hierarchical RNN encoder (Serban et al., 2016); (ii) SeqGAN: a GAN based model for sequence generation (Li et al., 2017b); (iii) CVAE baseline (Zhao et al., 2017); (iv) Dialogue WAE, a conditional Wasserstein auto-encoder for response generation (Gu et al., 2019); (v): A hierarchical VAE model (Serban et al., 2017). (vi) VHCR: a hierarchical VAE model with conversation modeling (Park et al., 2018). (vii) iVAE$_{\text{MI}}$: An implicit VAE model augmented with mutual information regularizer (Fang et al., 2019). The full comparison in shown in Table 12.

Stylized response generation: StyleFusion In this task, the additional sentences $\boldsymbol{b}$ are used to bias the generated response towards the reference style. The biased response representation is $\boldsymbol{z}'_{\tau} = \tau \boldsymbol{z}_{\text{Style}} + (1-\tau)\boldsymbol{z}_{\text{AE}}$, where $\tau \sim \text{Uniform}(0,1)$ and $\boldsymbol{z}_{\text{Style}}$ is the latent representation of $\boldsymbol{b}$. The corresponding loss for the biased target is $\mathcal{L}'_{\boldsymbol{x}} = -[\tau \log p(\boldsymbol{x}|\boldsymbol{z}_{\text{Style}}) + (1-\tau)\log p(\boldsymbol{x}|\boldsymbol{z}_{\text{AE}})]$, which is added into $\mathcal{L}_{\text{dialog}}$ for training.

Evaluation Two type of *Accuracy* are reported, based on text sequence (*i.e.,* neural) and its N-gram information. The accuracy is assessed by an oracle classifier to correctly predict whether generated response belongs the style-reference dataset.

Metric	LM	Representation		Learning Objective		
Method	PPL ↓	MI ↑	AU ↑	-ELBO ↓	KL ↑	Rec ↓
Ours($\lambda=0.05$)	22.34	5.34	32	282.70	6.97	282.84
Ours($\lambda=0.10$)	22.56	5.80	32	289.88	7.77	282.11
Ours($\lambda=0.25$)	22.63	7.42	32	290.69	11.19	279.49
Ours($\lambda=0.50$)	23.11	8.85	32	293.34	17.45	275.89
Ours($\lambda=1.0$)	24.92	9.18	32	301.21	30.41	270.80
GPT-2	22.00					
LSTM-LM	60.75			328.00		
LSTM-AE		9.26	32			278.76
SA-VAE	60.40	2.70	10	327.20	5.20	325.00
M. Annealing	61.21	0.0	0	328.80	0.0	328.80
C. Annealing	64.26	0.0	1	332.68	0.03	332.65
Aggressive	59.77	2.9	15	328.40	5.70	322.70
AE-BP ($\lambda=5$)	59.28	8.08	32	329.31	10.76	318.55

Table 10: Comparison on Yahoo dataset.

Metric	LM	Representation		Learning Objective		
Method	PPL ↓	MI ↑	AU ↑	-ELBO ↓	KL ↑	Rec ↓
Ours($\lambda=0.05$, PT)	13.47	3.49	32	33.08	3.92	29.17
Ours($\lambda=0.10$, PT)	13.48	4.65	32	33.45	5.44	28.01
Ours($\lambda=0.25$, PT)	14.08	7.22	32	35.04	9.79	25.25
Ours($\lambda=0.50$, PT)	16.67	8.89	32	38.50	16.35	22.14
Ours($\lambda=1.00$, PT)	29.63	9.20	32	47.35	28.96	18.39
GPT-2 (Radford et al., 2019)	20.24					
LSTM-LM	21.44					
LSTM-AE		9.18	32			
M. Annealing (Bowman et al., 2016)	21.50	1.42	2	33.07	1.42	31.66
C. Annealing (Fu et al., 2019)	21.62	2.33	4	33.25	2.36	30.89
Aggressive (He et al., 2019)	21.16	1.38	5	32.95	1.42	31.53
AE-BP ($\lambda=5$) (Li et al., 2019)	21.64	7.71	32	34.47	9.53	24.94

Table 11: Comparison on SNLI dataset. For LSTM-LM and GPT-2, we report the exact negative log likelihood.

C.2.1 Label-Conditional Text Generation

The goal of this task is to generate sentences conditioned on a given label. We consider a two-stage algorithm to adapt OPTIMUS for this task. First, we fine-tune a VAE language model on the downstream dataset, and freeze the model parameters. In another word, the latent space is fixed. Second, we build a conditional GAN for the latent space. Let's denote the latent vectors for ground-truth sentences as $\boldsymbol{z}_{\text{true}}$. We build a generator G to produce $\boldsymbol{z}_{\text{fake}} = G(\epsilon, y)$, where ϵ is the random noise, and y is the label. A discriminator D is trained simultaneously to distinguish $\boldsymbol{z}_{\text{true}}$ and $\boldsymbol{z}_{\text{fake}}$. The learning objectives for conditional GAN is:

$$\begin{aligned}&\min_G \max_D \mathcal{L}_{\text{cGAN}} \\ &= \mathbb{E}_{\boldsymbol{x},y\sim q(\boldsymbol{x},y)}\big[\mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z}|\boldsymbol{x})}[\log p_D(d=1|E(\boldsymbol{x}))] \\ &+ \mathbb{E}_{\epsilon\sim p_0(\epsilon)}[\log p_D(d=0|G(\epsilon,y))]\big] \end{aligned} \tag{16}$$

To make the model work effectively, it is key to learn a smooth and meaningful latent space of target sentences. The text generation procedure conditioned on label y is:

$$\boldsymbol{x} \sim p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}), \quad \text{with} \quad \boldsymbol{z} = G(\epsilon, y) \tag{17}$$

This mimics the process to produce the outlines of the sentences using conditional GAN, and fill in

details using the decoder. We show some generated sentences in Table 20.

We compare with three baselines: (1) *Ctrl-Gen* (Hu et al., 2017); We use their released code to reproduce the results. (2) *ARAE* (Zhao et al., 2018) proposes to learn an auto-encoder first, and then train a GAN to produce the latent vectors. (3) *NN-Outlines* (Subramanian et al., 2018) proposes the use of a general purpose encoder for text generation, and we implement it using BERT. Note that our two-stage fine-tuning scheme borrows the ideas from ARAE and NN-Outlines. The key difference is that we employ our pre-trained OPTIMUS model, and work on a better latent space.

Evaluation We consider three metrics: (1) *Bleu* for sentence quality, (2) *Accuracy* for conditional generation capability. The accuracy is assessed by an oracle classifier to correctly predict the attributes that generated sentences are conditioned on. (3) *G-score* is reported as the geometric mean of Accuracy and Bleu. This is the most important metric, as it evaluates the overall performance. For label-conditional text generation, Bleu of each generated sentence is computed by comparing with all sentences in the test set, as there are no source sentences. We further report Self-Bleu (Zhu et al., 2018) to evaluate the diversity of generated sentences.

C.3 Latent space interpolation & arithmetic operation

The latent space interpolation examples are shown in Table 13, 14, 15. The latent vector arithmetic operation examples are shown in Table 17, 18, 19.

C.4 Ablation study on VAE & AE objectives

We compare the interpolation examples in Table 16, and generally observe that VAE can produce smoother sentences interpolation results than AE. We compare the two pre-training objectives on the GLUE benchmark using the feature-based approach. The results are shown in Table 21. We see that both objectives outperform than BERT on large datasets, and VAE objective performs better than AE objective. This verifies the effectiveness of smooth regularization on the latent space for the classification performance.

Metrics	Seq2Seq	SeqGAN	CVAE	VHRED	VHCR	WAE	iVAE$_{\text{MI}}$	OPTIMUS
BLEU-Recall↑	0.232	0.270	0.265	0.341	0.271	0.289	0.355	**0.362**
BLEU-Precision↑	0.232	0.270	0.222	0.278	0.260	0.266	0.239	**0.313**
BLEU-F1↑	0.232	0.270	0.242	0.306	0.265	0.277	0.285	**0.336**

Table 12: Dialog response generation on `DailyDialog` dataset. All numbers are from (Gu et al., 2019) except that iVAE$_{\text{MI}}$ is from (Fang et al., 2019).

0.0	a young woman with a black hairbrush brushes her teeth while a man in a white shirt watches.
0.1	a blond woman with a black hairbrush brushes her teeth while a blond woman with a white hairbrush brushes her teeth.
0.2	a blond woman with a black hairbrush brushes her teeth while a man in a blue shirt watches.
0.3	a blond woman with a black hairbrush brushes her teeth while a man in a blue shirt watches.
0.4	a young woman in a blue shirt and blue jeans is lifting a large plastic bottle from a bottle.
0.5	a man in a blue shirt and blue jeans is brushing his teeth while a woman in a white shirt and blue pants looks on.
0.6	a man in a blue shirt is holding a small plastic bag while another man in a white shirt holds a large plastic bag.
0.7	a man in a blue shirt is holding a small plastic bag while another man in a white shirt holds a large plastic bag.
0.8	a man in a blue shirt is holding a bag of frozen peas while another man in a white shirt looks on.
0.9	a man in a blue shirt is holding a bag of food in a small bowl.
1.0	a man in a blue shirt is holding a bag of food in a small area of grass.

Table 13: Interpolating latent representation from plural sentence to singular sentence. Each row show τ and the sentence generated from the latent vector $\boldsymbol{z}_\tau$.

0.0	people are walking near a road.
0.1	people are walking near a bench.
0.2	people are sitting on a bench near a road.
0.3	people are sitting on a bench near a road.
0.4	some people are sitting on a bench outside.
0.5	there are two people sitting on a bench.
0.6	there are two men sitting on a bench waiting for a train.
0.7	there are two men sitting on a bench and looking at the sky.
0.8	there is a man sitting on the side of a boat.
0.9	there is a man sitting on the side of a boat and a woman sitting on the other side.
1.0	there is a man sitting on the side of a boat and the woman is sitting on the side of a boat.

Table 14: Interpolating latent representation from short sentence to long sentence. Each row show τ and the sentence generated from the latent vector $\boldsymbol{z}_\tau$.

0.0	i have been here a few times and i have never had a bad experience. i ordered the chicken and waffles. the chicken was cooked perfectly and the waffles were delicious. the waffles were also very good. i would definitely come back here again.
0.1	i have been going to this place for years. i had the chicken fried rice and it was delicious. the service was great and the food was fresh. i will definitely be back. i will definitely be back.
0.2	i have been going to this place for years. i was surprised to find out that they have a new location. the food is great and the service is great. i ordered the [UNK] chicken and it was delicious. i also ordered the [UNK] chicken and it was delicious. i will definitely be back.
0.3	i've been here a few times and it's always been great. the food is always fresh and the service is always fast. i'm not sure if they have a [UNK] or not but i'm sure they have a [UNK]. i'm sure they will be back soon.
0.4	i'm not sure what to say about this place. they have a great selection of food and drinks. i had the [UNK] and it was delicious. the staff was friendly and helpful. i will definitely be back.
0.5	i'm not sure what to say about this place. they have a great selection of food and the staff is very friendly. i'm not sure if they have a [UNK] or not. i'm sure they will be back soon.
0.6	wow! this place is awesome! they have a great selection of food and the staff is very friendly. i will definitely be back.
0.7	wow! this place is awesome! they have a great selection of food and the staff is very friendly. i will definitely be back.
0.8	great place! they have a great selection of food. they also have a great customer service. i will definitely be back!
0.9	great place! they have a great selection of products. they are very friendly and helpful. i will definitely be back!
1.0	great place! they have a great customer service. they are very friendly and helpful. they are also very helpful with the [UNK]. i will definitely be back!

Table 15: Interpolating latent representation within the same sentiment. Each row show τ and the sentence generated from the latent vector $\boldsymbol{z}_\tau$.

	OPTIMUS (VAE, $\beta = 1$)	**OPTIMUS (AE, $\beta = 0$)**
$\tau = 0.0$	the little girl plays with the toys.	the little girl plays with the toys.
$\tau = 0.1$	the child plays with the toy train.	the little girl plays the playground toy.
$\tau = 0.2$	the children play with a toy car.	the children play the miniature train ride.
$\tau = 0.3$	the children play in the ground.	the children play in the museum's playground.
$\tau = 0.4$	the children play in the playground	the children are watching a playhouse.
$\tau = 0.5$	the children are playing in the playground.	the children are watching a playhouse
$\tau = 0.6$	the children are watching a play.	the children are watching a playhouse
$\tau = 0.7$	the children are watching a show.	there are children watching a train.
$\tau = 0.8$	there are children watching a circus.	there are children watching a train.
$\tau = 0.9$	there are children watching a train.	there are children watching a train.
$\tau = 1.0$	there are children watching a train.	there are children watching a train.

Table 16: Comparison of VAE and AE objective for latent space interpolation. VAE shows smoother interpolation results than AE.

Source $\boldsymbol{x}_A$	**Target $\boldsymbol{x}_B$**
two soccer players are playing soccer	the people are building a machine
Input $\boldsymbol{x}_C$	**Output $\boldsymbol{x}_D$**
• people walking in the street	• the people were going to build the city
• the man was waiting for his wife to come home	• the man was going to get the job done
• two women preparing food for a table	• the people carefully prepared a piece of equipment
• two dogs chase each other through the water	• the vehicles get to work
• a person sitting in a library reading	• a person working on the building
• a tall human walking	• a construction project was made
• a young boy and a young girl play in a grassy field	• a child is building a house for the future to see
• men playing music in the rain	• they were making a construction work

Table 17: Sentence transfer via arithmetic operation in the latent space. The output sentences are in blue. In this example, we see content transition from *relaxing* to *working*.

Source $\boldsymbol{x}_A$	**Target** $\boldsymbol{x}_B$
a girl makes a silly face	two soccer players are playing soccer
Input $\boldsymbol{x}_C$	**Output** $\boldsymbol{x}_D$
• a girl poses for a picture	• two soccer players are at a soccer game.
• a girl in a blue shirt is taking pictures of a microscope	• two football players in blue uniforms are at a field hockey game
• a woman with a red scarf looks at the stars	• two men in white uniforms are field hockey players
• a boy is taking a bath	• two baseball players are at the baseball diamond
• a little boy is eating a bowl of soup	• two men are in baseball practice
• a mother is feeding her baby	• football players are at home
• a black dog is running across a field in the middle of a snowy field	• two white and black soccer players are in the field in a soccer field
• some dogs are traveling to their owners	• dogs are in the field playing baseball
• the men were sitting on the bench at the gym for a long time	• men on the field are playing in the league championship game

Table 18: Sentence transfer via arithmetic operation in the latent space. The output sentences are in blue. In this example, we see two type of style transition: (1) from singular to plural subject, and (2) from daily-life activity to sport.

Source $\boldsymbol{x}_A$	**Target** $\boldsymbol{x}_B$
people are walking near a road.	a girl is riding a small white horse in a park with a large group of people
Input $\boldsymbol{x}_C$	**Output** $\boldsymbol{x}_D$
• some people are holding cameras	• a girl in a black and white costume is performing a trick on a toy gun.
• people are attending church	• a young girl is participating in a martial arts competition in the middle of the night.
• people eat at a restaurant.	• a girl plays a `[UNK]` in a carnival in a city.
• the dancers are asleep	• the female ballet dancer is performing a ballet in the middle of a ballet class.
• two dogs are reunited	• a young girl is the first to capture a black and white dog in a black and white toy.
• a person is fishing for water.	• a girl is flying a kite into a tropical storm with a tropical storm.
• a mother and daughter laugh as they walk home	• a young blond-haired girl is rescued from a sad death by a young blond-haired girl in a karate ballet costume.
• a female gymnast is performing for a crowd	• a young girl is a solo performer in a karate ballet ballet performance in a ballet performance
• a small dog is in water	• a little girl is a golden retriever in a blue and white striped striped swimsuit

Table 19: Sentence transfer via arithmetic operation in the latent space. The output sentences are in blue. In this example, we see two type of style transition: (1) from plural/old to singular/young subject, or and (2) sentences are expended.

Positive
our favorite place to get great coffee and taterts.
the best brunch you will find in vegas.
the best breakfast with meats is awesome!
great samosas and serve as a regular!
great place to meet up with a custom bean & wine.
a great selection of chinese food and always happy.
the free wi-fi is amazing as well!
great staff and freshly made latte is a must.
love the fresh staff as well!
highly recommend the place and sunbeams!
the staff is always great with homemade paesadillas.

Negative
not only did you get a headache upstairs, they were disgusting.
once i realized the pizza wasn't decent, i cancelled.
instead of going to the bathroom you couldn't find anything.
tonight i was unable to give the pizza any less.
i didn't even bother to find a $ [num] frozen pizza.
no wonder i was dropped off at laundry.
not only was this place freezing, but the salad sucked.
then [num] bucks was ruined in my mouth.
love the fresh staff as well!
once you asked for chipotle its out of control.
another thing i refused to eat.

Table 20: Label-conditional text generation on Yelp dataset. The top block shows the positive reviews, and bottom block shows the negative reviews.

System		MNLI	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	WNLI	**Average**
Dataset size		392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	634	
Feature-based	BERT	0.414	0.146	0.673	0.731	0.187	0.690	0.812	0.549	0.577	0.531± 0.011
	OPTIMUS (VAE)	0.468	0.662	0.720	0.789	0.144	0.719	0.816	0.585	0.563	**0.607**± 0.013
	OPTIMUS (AE)	0.442	0.565	0.692	0.788	0.046	0.655	0.812	0.498	0.620	0.569± 0.010
Fine-tuning	BERT	0.835	0.909	0.912	0.923	0.598	0.886	0.868	0.700	0.507	0.793 ± 0.008
	OPTIMUS (VAE)	0.834	0.909	0.908	0.924	0.573	0.888	0.873	0.697	0.563	0.798 ± 0.017

Table 21: Ablation study on the AE and VAE objective of OPTIMUS. Comparison is on the validation set of GLUE. F1 scores are reported for QQP and MRPC, Spearman correlations are reported for STS-B, and accuracy scores are reported for the other tasks.