MSc Dissertation

Understanding the Latent Space in Variational Autoencoders

Lan Zhang
Supervised by Dr Ehsan Shareghi

Department of Electrical and Electronic Engineering
Sepetember 2020

Abstract

Data representation is one of the most important problems in machine learning. Better representation of data can bring more reasonable outcome of machine learning algorithms. Variational Autoencoder (VAE) has shown powerful ability to address this problem in an unsupervised fashion. However, while successful, for text data it is not clear which aspects of the data are captured by the latent code in VAEs. This research focuses on understanding the learned representations in VAEs for text. We implement several quantitative experiments and qualitative experiments on natural text and two artificially created datasets. We find that VAEs prefer to encode the first few words of sentences into the latent code. Moreover, the structure information of sentences is also captured by VAEs.[1]

[1]Code available at: https://github.com/LanZhang-UCL/dissertation

Contents

List of Figures

List of Tables

List of Abbreviations

AE Autoencoder.

AU Active Unit.

CNN Convolutional Neural Network.

ELBO Evidence Lower Bound.

IPHR Individual Position Hit Rate.

LR Logistic Regression.

LSTM Long Short-Time Memory.

MLP Multi-Layer Percepton.

ND Noise Dimension.

NLP Natural Language Processing.

OPHR Overall Position Hit Rate.

POS Part-of-Speech.

RNN Recurrent Neural Network.

SD Signal Dimension.

SVM Support Vector Machine.

VAE Variational Autoencoder.

1 Introduction

Data representation is an important problem in machine learning. Machine learning algorithms such as Logistic Regression (LR), Support Vector Machine (SVM) and Multi-Layer Percepton (MLP) require features extracted from data to do inference. In this context, features can represent the data. In the past, feature engineering was needed to extract features manually from original data. In the era of deep learning, features can be extracted automatically through deep neural networks so that the original data can be fed into the model directly to do inference. Typically, the original data has a high dimension, hence on the original space, training the model will be computational inefficient. In addition, the original data usually does not highlight the information needed for tasks. The important information of the same data for different tasks is usually various. Hence, it is important to find a suitable data representation in order to compress data without losing important information.

Learning data representation can also help to analyse and understand the characteristics of data itself. Normally there are two categories of machine learning in Natural Language Processing (NLP): supervised learning and unsupervised learning. Supervised learning is used to deal with tasks whose data has been labelled. For example, in tasks of detecting sentiment information of sentences, sentences can be labelled with three labels: positive, negative and neutral, and a relevant supervised learning task is to classify a sentence. In contrast to supervised learning, unsupervised learning is used to cope with data without label. Therefore, unsupervised learning focuses more on data itself rather than some labels assigned to the data. Supervised representation learning techniques are useful to find a data representation, but they still need labelled data to extract label-relevant features. Unsupervised representation learning algorithms are great alternatives since they do not require data to be labelled.

Autoencoder (AE) [2] framework is a successful unsupervised learning method for data representation. The framework of an autoencoder is illustrated in Figure 1. The encoder uses a deterministic or stochastic function φ_{enc} to determine the representation code $\mathbf{z}$ based on the datapoint $\mathbf{x}$. The decoder is another function which reconstructs original datapoint $\mathbf{x}$ based on the code $\mathbf{z}$. The encoder is used to transform the original data to the representation. Passing the representation to relevant decoder, decoder can use the representation to reconstruct the original data. As an autoencoder does not need extra annotation for data, this learning representation method is an unsupervised learning method.

Replacing the encoder and decoder with a stochastic neural encoder and decoder, $q_\phi(\mathbf{z}|\mathbf{x})$ and $p_\theta(\mathbf{x}|\mathbf{z})$, and optimising the neural networks by maximising the Evidence Lower Bound (ELBO) will turn an Autoencoder into a Variational Autoencoder (VAE) [3], which treats representation learning as an inference problem. In addition to compressing data and learning representation, VAE can be used to generate data, and have shown success in generating several kinds of data such as music [4], speech and handwriting [5], arithmetic expressions and molecular structures [6], and sentences [1]. When applied to generate text data, VAE can provide grammatically correct sentences whereas AE cannot promise to do this.

Table 1 shows this behaviour. Sentences in AE are not meaningful whereas sentences in VAEs are. Despite the success of VAEs on text, few works focus on what kinds of information are captured exactly

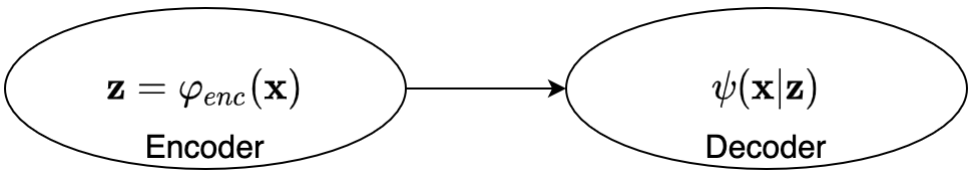


Figure 1: The framework of Autoencoders.

Autoencoder	Variational Autoencoder
1. if such just captain and those another such became old; – whitefoot the track what feast. 2. if such jumper has those frog in this tense song; whenever a deer at which place. 3. now the voice that had lived in the wild that beside earth; and his life will bring. 4. like him that he had by the beautiful soft – which he flies after his adventure comes when brothers. 5. like him that the hunter was in the sea beside the bride; which does <unk> when they ye could. 6. -rsb- least twice the man occupied a very <unk> through the earth; all who could call whither the brothers.	1. sometimes i think, if i can, and i ll go and see if i can go and see the <unk>, and i ll be back. 2. my dear boy, i do n t want to go back to the <unk>, and i ll go and see. 3. no, i never could, but i could see him, and he was n t <unk>. 4. i know i do n t want to go back in the morning. 5. i do n t believe they re not a bit. 6. i do n t believe the way.

Table 1: Sentences in Autoencoder and Variational Autoencoder. Sentences are obtained by using latent codes to decode sentences. The latent codes of the first and last sentence in both columns are obtained by randomly sampling two latent code using standard Gaussian distribution and the latent codes of others are intermediate codes of two sampled latent code.

by VAEs which make sentences grammatically correct. Although the performance is the main focus in machine learning, understanding behaviour of the model is also important.

The main step of understanding VAEs is to understand the latent space where the hidden code lies on. In this dissertation, we analyse VAEs for text and aim to obtain a better understanding of the behaviour of VAEs. To this end, we first evaluate VAEs on natural text and find some patterns. Then we carefully design two toy datasets which incorporate certain information in the data to support findings. More concretely: we find a better way to measure the activity of a dimension in latent space; to some degree, we figure out which aspects of natural text are captured by VAEs; we evaluate how sentences change with the changes of latent code; we design two toy datasets which contain certain information and evaluate VAEs on these two toy datasets.

1.1 Thesis Outline

The rest of this dissertation is organized as follows: chapter 2 introduces the essential background of this work and link this work to other related works; chapter 3 describes experiments on natural text; chapter 4 explains the details of toy dataset 1 and present implementation of relevant experiments; chapter 5 discusses how to construct toy dataset 2 and results of relevant experiments; chapter 6 concludes this work and suggests the directions of future work.

2 Background Theory and Literature Review

This chapter will introduce some necessary background for this research, including the concept of VAEs, the optimization challenge of training VAEs, the architecture of VAEs for text, and the pattern of dimensions in the latent space of VAEs.

2.1 Variational Autoencoders

The variational autoencoders [3] are derived from autoencoders. Let $\mathbf{x}$ denote datapoints in data space and $\mathbf{z}$ denote latent variables or hidden code in the latent space. The assumption of VAEs is such that the datapoint in the data space is generated by the combination of two random process. The first random process is to sample a point $\mathbf{z}^{(i)}$ from the latent space in VAEs with a prior distribution of $\mathbf{z}$, denoted by $p(\mathbf{z})$. The second random process is to generate a point $\mathbf{x}^{(i)}$ from the data space, denoted by $p(\mathbf{x}|\mathbf{z}^{(i)})$. VAE uses a combination of a probabilistic encoder $q_\phi(\mathbf{z}|\mathbf{x})$ and a probabilistic decoder $p_\theta(\mathbf{x}|\mathbf{z})$ to learn this statistical relationship between $\mathbf{x}$ and $\mathbf{z}$. The parameters of encoder and parameters of decoder are denoted by ϕ and θ respectively. This framework of VAEs is shown in Figure 2.

2.1.1 Objective Function

The objective function of VAEs is:

$$\mathcal{L}(\phi, \theta; \mathbf{x}) = -\mathrm{KL}(\mathrm{q}_\phi(\mathbf{z}|\mathbf{x})||\mathrm{p}(\mathbf{z})) + \mathbb{E}_{\mathrm{q}_\phi(\mathbf{z}|\mathbf{x})}[\log(\mathrm{p}_\theta(\mathbf{x}|\mathbf{z}))] \tag{1}$$

which is a lower bound of the logarithmic data distribution $\log p(\mathbf{x})$, called evidence lower bound (ELBO). The first term of objective function is a Kullback-Leibler (KL) divergence term which measures the difference between posterior distribution $q_\phi(\mathbf{z}|\mathbf{x})$ and prior distribution $p(\mathbf{z})$ and can be seen as a regularisation term. The second term is reconstruction loss, which is the expectation of the logarithm of marginal likelihood based on the posterior distribution modelled by the encoder. VAEs learn parameters ϕ and θ by maximising ELBO.

2.1.2 KL Divergence Term and Reparameterization Trick

Due to mathematical convenience, the multivariate Gaussian distribution with zero mean and unit variance (i.e., $\mathcal{N}(\mathbf{0}, I)$) is used for prior distribution $p(\mathbf{z})$ and the learned conditional posterior distribution $q_\phi(\mathbf{z}|\mathbf{x})$ is usually assumed to be a class of diagonal multivariate Gaussian distribution (i.e., $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Lambda})$, in which $\boldsymbol{\Lambda} = (\sigma_1^2, \sigma_2^2, ..., \sigma_n^2)$ is a diagonal positive definite matrix). By this setting, the KL divergence term in Eq. 1 can be computed in a closed-form expression:

$$\mathrm{KL}(\mathrm{q}_\phi(\mathbf{z}|\mathbf{x})||\mathrm{p}(\mathbf{z})) = \frac{1}{2}\sum_{\mathrm{i}=1}^{\mathrm{n}}(\mu_\mathrm{i}^2 + \sigma_\mathrm{i}^2 - \log\sigma_\mathrm{i}^2 - 1) \tag{2}$$

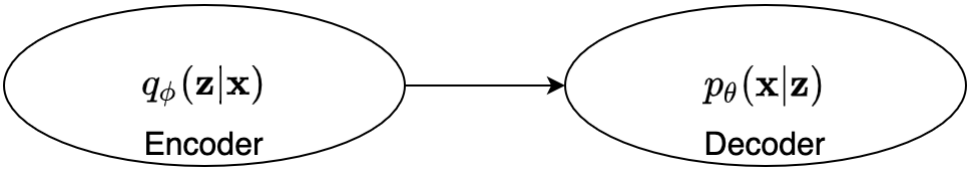


Figure 2: The framework of Variational Autoencoders.

The proof of this closed-form expression is provided in Appendix A.1.

The reconstruction loss in Eq. 1 does not have a closed-form expression. Therefore, Monte Carlo method [7] is often used to estimate the reconstruction loss. Monte Carlo method is to calculate the expectation by sampling L points from variables space using the distribution and calculating the average of values on these several points. Using Monte Carlo method, the reconstruction loss is estimate as follows:

$$\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log(p_\theta(\mathbf{x}|\mathbf{z}))] = \frac{1}{L}\sum_{l=1}^{L}\log p_\theta(\mathbf{x}|\mathbf{z}^{(l)})) \tag{3}$$

However, the sampling process does not have gradient which causes the difficulty of using gradient-based algorithms to optimise objective function. The reparameterization trick [3] can solve this problem and simplify the sampling process in Mento Carlo method. The reparameterization trick expresses the random variable $\mathbf{z}$ in posterior distribution $q_\phi(\mathbf{z}|\mathbf{x})$ as this:

$$\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, I) \tag{4}$$

Instead of sampling $\mathbf{z}^{(l)}$ using distribution $q_\phi(\mathbf{z}|\mathbf{x})$, sampling $\epsilon^{(l)}$ using $\mathcal{N}(\mathbf{0}, I)$ and computing $\mathbf{z}^{(l)}$ using Eq. 4 will bring the same result. Through this trick, the random process is a unified random process which does not change with datapoints. Datapoints only affect computing not sampling.

2.2 Posterior Collapse

A common optimisation challenge of training VAEs in text modelling is called posterior collapse such that the learned posterior distribution $q_\phi(\mathbf{z}|\mathbf{x})$, is very close to the prior distribution $p(\mathbf{z})$ which makes the KL divergence term near zero. Posterior collapse indicates that the posterior distribution is not conditioned on $\mathbf{x}$. Therefore, when posterior collapse happens, the encoder does not encode any useful information about data into the latent variables $\mathbf{z}$ and the decoder ignores the latent variables. Several strategies could be used to alleviate posterior collapse from different methods. One approach is to change the architecture of VAEs such as leveraging weak decoders (compare with strong decoder) like Convolutional Neural Network (CNN) decoders [8] or making additional connections between encoder and decoder [9]. Another method is to adjust the training process. Strategies using this method include KL cost annealing and word dropout [1], training the encoder and the decoder asynchronously [10]. Posterior collapse can also be prevented by adding constraints to the KL term. Because posterior collapse happens when KL term is close to zero, an intuitive approach to prevent posterior collapse is to control the impact of the KL term explicitly. This can be done by δ-VAE [11] which constrains the KL term not lower than some threshold δ by changing latent variables to the last element of a sequence of latent variables, or β-VAE [12] which adds an additional parameter to the KL divergence term of original objective function Eq. 1.

In this research, to prevent posterior collapse, instead of using ELBO as the objective function, we use an extension of β-VAE [13]:

$$\mathcal{L}(\phi, \theta; \mathbf{x}) = -\beta\,|\mathrm{KL}(\mathrm{q}_\phi(\mathbf{z}|\mathbf{x})||\mathrm{p}(\mathbf{z})) - \mathrm{C}| + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log(p_\theta(\mathbf{x}|\mathbf{z}))] \tag{5}$$

where C is a positive real value which represents the target KL divergence term value. We set $\beta = 1$ to make sure the weights of two terms balanced. This can be seen as a constraint optimisation, where β acts as an Lagrange Multiplier [14].

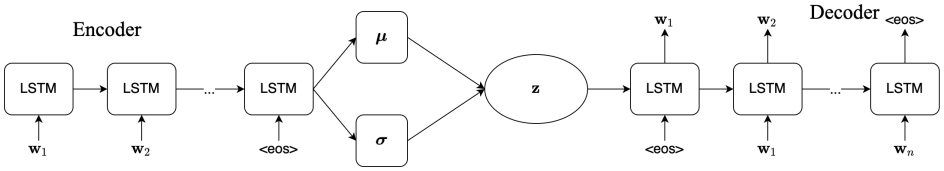


Figure 3: The architecture of a VAE for sentences [1].

2.3 Model Architecture

Recurrent Neural Network (RNN) is a neural network architecture for sequence data. Using $\mathbf{x}_t$, $\mathbf{h}_t$ and $\mathbf{y}_t$ to denote the input, hidden state and output of RNNs at timestamp t respectively, a typical RNN network contains three neural network layers such that:

$$\mathbf{h}_t = f(\mathbf{W}_{in}\mathbf{x}_t + \mathbf{W}_h\mathbf{h}_{t-1} + \mathbf{b}_1) \tag{6}$$

$$\mathbf{y}_t = g(\mathbf{W}_{out}\mathbf{h}_t + \mathbf{b}_2) \tag{7}$$

where $\mathbf{W}_{in}$, $\mathbf{W}_h$ and $\mathbf{W}_{out}$ are the weights of three layers, and $\mathbf{b}_1$ and $\mathbf{b}_2$ are the biases. Long Short-Time Memory (LSTM) [15] is a kind of RNN which addresses the issue of gradient backpropagation explosion or vanishing in typical RNN by changing the hidden layer from neurons to LSTM cells with a new hidden state $\mathbf{s}_t$. The working principle of LSTM is such that:

$$\mathbf{g}_t^{(in)} = f_{in}(\mathbf{W}_x^{(in)}\mathbf{x}_t + \mathbf{W}_s^{(in)}\mathbf{s}_{t-1} + \mathbf{W}_h^{(in)}\mathbf{h}_{t-1} + \mathbf{b}^{(in)}) \tag{8}$$

$$\mathbf{s}_t = \mathbf{s}_{t-1} + g_{cell}(\mathbf{W}_x^{(cell)}\mathbf{x}_t + \mathbf{W}_s^{(cell)}\mathbf{s}_{t-1} + \mathbf{b}^{(cell)}) \odot \mathbf{g}_t^{(in)} \tag{9}$$

$$\mathbf{g}_t^{(out)} = f_{out}(\mathbf{W}_x^{(out)}\mathbf{x}_t + \mathbf{W}_s^{(out)}\mathbf{s}_t + \mathbf{W}_h^{(out)}\mathbf{h}_{t-1} + \mathbf{b}^{(out)}) \tag{10}$$

$$\mathbf{h}_t = h_{cell}(\mathbf{s}_t) \odot \mathbf{g}_t^{(out)} \tag{11}$$

where $\mathbf{g}_t^{(in)}$ and $\mathbf{g}_t^{(out)}$ are called input gate and output gate. A forget gate [16] can also be applied to the original LSTM by adding:

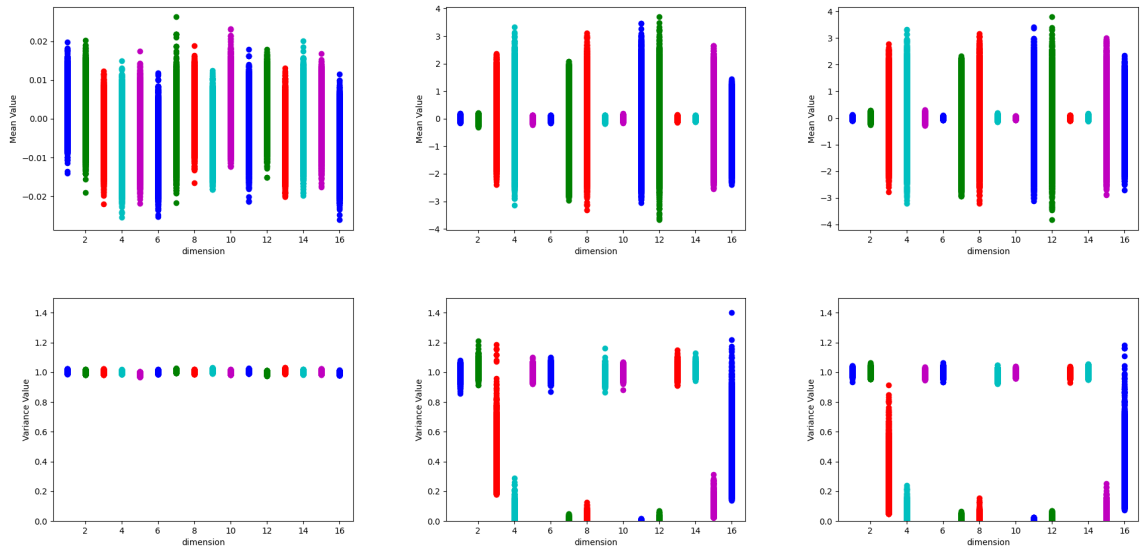
$$\mathbf{g}_t^{(forget)} = f_{forget}(\mathbf{W}_x^{(forget)}\mathbf{x}_t + \mathbf{W}_s^{(forget)}\mathbf{s}_{t-1} + \mathbf{W}_h^{(forget)}\mathbf{h}_{t-1} + \mathbf{b}_{(forget)}) \tag{12}$$

and modifying:

$$\mathbf{s}_t = \mathbf{g}_t^{(forget)} \odot \mathbf{s}_{t-1} + g_{cell}(\mathbf{W}_x^{(cell)}\mathbf{x}_t + \mathbf{W}_s^{(cell)}\mathbf{s}_{t-1} + \mathbf{b}_{cell}) \odot \mathbf{g}_t^{(in)} \tag{13}$$

The LSTM with input gate, forget gate and output gate is the commonly used LSTM. The cell activation functions g_{cell}, h_{cell} usually choose $\tanh$ and the gate activation functions f_{in}, f_{out}, f_{forget} usually choose logistic function.

We leverage LSTMs in VAEs. The architecture of a VAE for sentences [1] is depicted in Figure 3. $\mathbf{w}_1$, $\mathbf{w}_2$, ..., $\mathbf{w}_n$ are words of a sentence. The subscript represents the position of word in the sentence. We use the standard multivariate Gaussian distribution for prior distribution $p(\mathbf{z})$ and diagonal multivariate Gaussian distributions for posterior distributions $q_\phi(\mathbf{z}|\mathbf{x})$. $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are the mean and the square root of variance of posterior distributions. "<eos>" is a special word to indicate the end of a sentence which also indicates the start to decode sentence in the decoder. We concatenate the latent code with word embedding at every timestamp as the input of the decoder. During training, we use the words of original sentences for the decoder's input rather than the predicted words from the decoder. After training, when decoding or reconstructing sentences from the latent code, we use greedy decoding, which chooses the word with the highest probability at each timestamp.



(a) before training (b) during training, epoch=8 (c) end training, epoch=16

Figure 4: VAE (C=15, dim=16), best run (i.e., model with smallest reconstruction loss). The top row and bottom row represents the mean and variance of posterior distributions of sentences in validation set respectively. In this figure, colors are only used to make a distinction between neighbour dimensions and mean nothing.

2.4 Active Units in the Latent Space

A common behaviour in VAEs is such that not all dimensions carry information about the original data. Exploring those dimensions which do not carry information is not meaningful. Therefore, it is important to filter those dimensions to reduce the dimensionality of scope. One method of measuring the activity of a dimension is Active Unit (AU) [17]. A dimension u is defined to be active if the statistic $A_u = \mathrm{Cov}_{\mathbf{x}}(\mathbb{E}_{\mathrm{u}\sim\mathrm{q}(\mathrm{u}|\mathbf{x})}[\mathrm{u}])$ is larger than 0.01, where Cov denotes covariance. The definition of AU suggests that if the changes of the mean value of posterior distribution on this dimension across all data are significant, this dimension is used to carry information. While effective, when using diagonal Gaussian distributions as posterior distributions, this concept only focuses on the change of mean of posterior distributions on this dimension and ignores the variance of posterior distributions. VAEs require both mean and variance of posterior distributions during training whereas AEs only require isolated latent codes which can be considered as mean. Focusing on variance rather than mean might be more effective when evaluating dimensions of the latent space in VAEs.

2.5 An Alternative to Active Units

To intuitively show how latent space changes during training, we choose a model and plot the mean and variance of posterior distributions of sentences in validation set during training in Figure 4. We observed a phenomenon which occurred in all VAEs such that before training, all posterior distributions are close to the prior distribution whose mean is zero and variance is one. However, during training posterior distributions on some dimensions are moving away from prior distribution which causes larger range of mean value, and a variance value which is not always close to one, whereas posterior distributions on other dimensions are still close to the prior distribution. Those dimensions on which posterior distributions are always close to the prior distribution are "**Noise Dimension (ND)**" because these dimensions do not carry the information from data and will confuse decoder, whereas other dimensions are "**Signal**

Dimension (SD)" because these dimensions carry the information from data. When training ends, the separation between signal dimensions and noise dimensions becomes more distinguishable.

We use the communication theory to interpret signal dimensions and noise dimensions. The encoder is the transmitter, the decoder is the receiver and each dimension in the latent space of VAE can act as a channel in communication. The transmitter and receiver are connected by channels. Transmitter chooses channels to transmit message. The unoccupied channels provide noise. The receiver does not know which channels are occupied to transmit message so that it should use the received signal from all channels to decode the message. The amount of information which should be transmitted in this system is constrained by a presetting value C. The receiver aims to achieves the lowest error rate. The error rate is relevant to the ability of receiver to distinguish signal and noise and decode the signal.

We focus on the change of variance and classify the i-th dimension as a noise dimension if:

$$\max_{\mathbf{x} \in X_{test}} \sigma_i^2 - \log \sigma_i^2 - 1 \leq \epsilon \tag{14}$$

where X_{test} is the test datatset, σ_i^2 is the variance of posterior distribution on the i-th dimension for a sentence $\mathbf{x}$ and ϵ is a threshold. This expression of σ is derived from the KL divergence term. Eq. 2 shows that the overall KL divergence term can be divided based on dimensions. KL divergence on the i-th dimension consists of two terms:

$$\mathrm{KL_i}(\mathrm{q_\phi}(\mathbf{z}|\mathbf{x})||\mathrm{p}(\mathbf{z})) = \frac{1}{2}(\mu_\mathrm{i}^2 + \sigma_\mathrm{i}^2 - \log \sigma_\mathrm{i}^2 - 1) \tag{15}$$

The first term is related to the mean of posterior distribution, and the second term is related to the variance and is used to classify dimensions. We suggest using $0.05 \log C$ as the threshold. It should be noted that changing the threshold of active units might bring similar results, however, it is difficult to find that threshold for different VAEs while in our method this becomes more systematic. We highlight the difference between techniques in our experiments.

This chapter has provided the theoretical knowledge and implementation of VAEs as well as techniques to prevent posterior collapse and methods to evaluate dimensions of the latent space. In the next three chapters, we experiment with VAEs on four datasets and report our findings in terms of the phenomenon captured by the latent code.

3 Experiments with VAEs on Natural Text

We trained VAEs on CBT (Children's Book Test) and Wiki datasets [18]. The (train, validation, test) set of CBT and Wiki contain (191802, 9690, 12280) and (2159541, 269917, 270403) sentences. The vocabulary of CBT dataset has 12000 words and the vocabulary of Wiki dataset has 19999 words. The unknown words in sentences are represented by a special word "<unk>". In this chapter, we leveraged 256 dimensions for word embedding and 512 dimensions for LSTMs for both encoder and decoder. We trained VAEs with 64-dimension latent space on CBT and Wiki, and 16-dimension latent space on CBT only. We set C to 3, 15, 50, 100, 200. We trained all models from 3 random starts. We used Adam [19] with learning rate 0.00075 to optimise objective function. We stopped training when the loss on the validation set did not decrease within 3 epochs, or the difference of loss on the training set within 3 epochs was smaller than 0.001. We set the maximum number of epochs to 50. We also trained two 64-dimension latent space VAEs with posterior collapse by setting C to 0 on CBT and Wiki as the baseline and three AEs with same numbers of dimensions as VAEs.

We test the performance of models on test set. We report the reconstruction loss and KL divergence on test set in Table 2. The results are the average of all data in the test set. We also report the number of signal dimensions and active units here. Similar to results in [18], the KL divergence is very close to C and with the increase of KL divergence term, the reconstruction loss decreases. However, when C exceeds some value, the reconstruction loss start increasing, and the standard deviation of reconstruction loss also increases. This happens for both 16-dimension models and 64-dimension models. In addition, because of variation, VAEs cannot perform as well as autoencoders, even with large enough C value. To

$\mathbf{z}$-dim=64	Rec. Loss	KL divergence	SD	AU
VAE (collapse, CBT)	64.82	0.02	-	-
VAE (C=3, CBT)	61.89(0.02)	3.16(0.02)	3.3(0.5)	3.3(0.5)
VAE (C=15, CBT)	52.75(0.03)	15.10(0.04)	7.0(0.0)	11.0(1.6)
VAE (C=50, CBT)	34.85(1.15)	50.10(0.10)	22.0(0.8)	44.0(2.8)
VAE (C=100, CBT)	22.93(3.27)	100.02(0.32)	39.0(4.2)	60.7(4.7)
VAE (C=200, CBT)	26.49(7.07)	199.90(0.22)	64.0(0.0)	63.3(0.5)
Autoencoder (CBT)	9.11	-	-	-
VAE (collapse, Wiki)	83.34	0.01	-	-
VAE (C=3, Wiki)	80.31(0.05)	3.22(0.02)	3.3(0.5)	3.3(0.5)
VAE (C=15, Wiki)	70.03(0.07)	15.12(0.01)	9.3(0.5)	9.3(0.5)
VAE (C=50, Wiki)	42.86(0.11)	50.26(0.04)	34.67(2.4)	35(2.2)
VAE (C=100, Wiki)	18.36(0.45)	100.31(0.15)	64.0(0.0)	64.0(0.0)
VAE (C=200, Wiki)	5.40(0.82)	200.01(0.12)	64.0(0.0)	64.0(0.0)
Autoencoder (Wiki)	3.46	-	-	-
$\mathbf{z}$-dim=16	Rec. Loss	KL divergence	SD	AU
VAE (C=3, CBT)	61.76(0.07)	3.16(0.02)	2.7(0.5)	2.7(0.5)
VAE (C=15, CBT)	52.52(0.07)	15.08(0.02)	7.0(0.8)	9.0(0.8)
VAE (C=50, CBT)	37.63(0.31)	50.07(0.00)	16.0(0.0)	16.0(0.0)
VAE (C=100, CBT)	43.76(3.31)	100.06(0.10)	16.0(0.0)	16.0(0.0)
VAE (C=200, CBT)	51.43(6.27)	199.84(0.17)	16.0(0.0)	16.0(0.0)
Autoencoder (CBT)	29.13	-	-	-

Table 2: Results on the test set. Rec. represents reconstruction loss. SD represents the number of signal dimensions. AU represents the number of active units. For models with several runs, we report the mean and (standard deviation).

$\mathbf{z}$-dim=64	Rec. Loss	BLEU-2/4 (Mean Vector)	BLEU-2/4 (Signal Vector)
VAE(C=3, CBT)	61.86(0.02)	9.08(0.55)/1.40(0.21)	9.00(0.56)/1.38(0.20)
VAE(C=15, CBT)	52.69(0.04)	14.76(0.31)/4.81(0.18)	14.44(0.36)/4.66(0.22)
VAE(C=50, CBT)	34.85(1.14)	28.03(0.75)/17.30(0.91)	27.51(0.77)/16.89(0.93)
VAE(C=100, CBT)	22.92(3.23)	39.88(3.11)/30.34(3.52)	39.40(3.03)/29.87(3.43)
VAE(C=200, CBT)	26.48(7.06)	39.04(4.29)/28.32(5.41)	39.04(4.29)/28.32(5.41)
VAE(C=3, Wiki)	80.17(0.05)	11.59(0.56)/3.29(0.22)	11.57(0.55)/3.29(0.21)
VAE(C=15, Wiki)	69.85(0.07)	14.70(0.33)/5.49(0.18)	14.69(0.33)/5.48(0.18)
VAE(C=50, Wiki)	42.71(0.11)	26.36(0.73)/17.03(0.64)	26.32(0.76)/17.00(0.66)
VAE(C=100, Wiki)	18.35(0.45)	51.35(0.99)/44.61(1.18)	51.35(0.99)/44.61(1.18)
VAE(C=200, Wiki)	5.40(0.82)	66.89(2.88)/62.56(3.49)	66.89(2.88)/62.56(3.49)
$\mathbf{z}$-dim=16	Rec. Loss	BLEU-2/4 (Mean Vector)	BLEU-2/4 (Signal Vector)
VAE(C=3, CBT)	61.74(0.09)	8.30(0.34)/1.44(0.06)	8.30(0.34)/1.44(0.06)
VAE(C=15, CBT)	52.53(0.08)	14.56(0.73)/4.76(0.37)	14.40(0.79)/4.68(0.40)
VAE(C=50, CBT)	37.61(1.39)	23.26(0.29)/12.98(0.31)	23.26(0.29)/12.98(0.31)
VAE(C=100, CBT)	43.76(3.31)	23.30(1.02)/12.25(1.55)	23.30(1.02)/12.25(1.55)
VAE(C=200, CBT)	51.43(6.27)	20.10(2.81)/8.24(3.03)	20.10(2.81)/8.24(3.03)

Table 3: The reconstruction loss of using signal vector and BLEU-2/4 scores of using mean of posterior distribution and signal vector of mean on test set. We report the mean and (standard deviation) here.

further evaluate these trained VAEs, we first focus on signal dimensions in the next section.

3.1 The Importance of Signal Dimensions

As Table 2 shown, higher C value encourages more signal dimensions and active units. This is because larger KL divergence term indicates higher amount of information needs to be transmitted and the transmitter should occupy more channels. We use this behaviour to interpret the behaviour of reconstruction loss. We treat the reconstruction loss as some kind of error rate. Because larger KL divergence encourages more signal dimensions and less noise dimensions, the signal-to-noise ratio (SNR) increases. As a well-known rule in the communication theory, larger SNR can bring lower error rate. When the constrained amount of information exceeds the capacity of the system, the encoder fails to learn the effective code. Comparing the number of signal dimensions and active units, we find that the number of signal dimensions is always less than or equal to the number of active units, except for VAE (C=200, dim=64, CBT). In fact, for all runs of all models, except one run of VAE (C=200, dim=64, CBT), signal dimensions are all active units.

To demonstrate that signal dimensions carry most information of the original data, we do a quantitative test. We mask noise dimensions and calculate the reconstruction loss again. The results are shown in Table 3. Compared to results in Table 2, the reconstruction loss does not have a significant change. The general trend is that the reconstruction loss has a slight decrease, which indicates better performance. This demonstrate that signal dimensions have encoded most information whereas noise dimensions have not encoded much information. Otherwise, removing noise dimensions should have caused an increase on reconstruction loss.

Furthermore, we mask noise dimensions of a vector in latent space and call the new vector as signal vector. We use the mean of the posterior distribution of a sentence in test set as the latent vector and the signal vector of mean to reconstruct test set separately. We evaluate the reconstruction using BLEU [20] score, which measures the similarity between candidate sentences and reference sentences. The results

	Sentence 1	Sentence 2	Sentence 3
sentence	i do n t expect to go.	after all, i d rather love you than not, hurt as it will.	at last he came to one of the largest forests in all the world, composed entirely of <unk>.
Mean	i do n t want any good.	and i feel as if i could n t say anything about it either.	at last he came to the top of the mountain, and he saw a troop of <unk> <unk>.
Active units	i do n t want any good.	and i feel as if i could n t say anything about it either.	at last he came to the top of the mountain, and he saw a troop of <unk> coming.
Signal Vector	i do n t want any good.	and i feel as if i could n t say anything about it either.	at last he came to the top of the mountain, and he saw a troop of <unk> coming.

Table 4: The reconstructed sentences of three sampled sentences in VAE (C=15, dim=64, CBT, best run).

are also shown in Table 3. Compared to the results of using latent code, the BLEU scores of using signal vector only have a slight decrease. This demonstrates that signal dimensions carry the information from the original data. If noise dimensions also carry the information from the original data, ignoring noise vector would have caused a significant drop in BLEU scores. We attribute this slight decrease to the fact that the decoder cannot completely ignore the noise produced by noise dimensions. The noise dimensions still have an influence on the reconstruction to some degree.

This section has shown that signal dimensions carry the information from the original data, however, this does not make the method of signal dimensions different to the method of active units. In the next section, we compare signal dimensions and active units to demonstrate that the method of signal dimensions is a better indicator.

3.2 Comparison between Signal Dimensions and Active Units

We do a qualitatively test to compare signal dimensions and active units. We sample two sentences from test set and feed them to the encoder of VAE (C=15, dim=64, CBT, best run). We choose this VAE because it has a proper number of signal dimensions, and some dimensions of this model are active units but not signal dimensions. We sample 3 sentences from test set and use the mean of posterior distributions, active units of mean and signal vector of mean to reconstruct sentences. The results are shown in Table 4. The reconstructed sentence from the mean takes some format of the original sentence. Although in one case the signal vector does not provide the same sentence as the mean, using active units and signal vectors to reconstruct sentences always has the same results. Because the number of signal dimensions is smaller than the number of active units, signal dimension method is more effective than active units.

We then mask one dimension of mean to show that signal dimension is more accurate compared to active unit. The results are shown in Table 5. When we drop a dimension which is both a signal dimension and an active unit, in most cases the reconstruction sentences change from the mean reconstructed sentence. However, when we drop a dimension which is an active unit but not a signal dimension, the reconstruction sentences do not have any difference to the mean reconstructed sentence (see the red text in Table 5). This indicates that some active units can be ignored for all sentences, but signal dimensions cannot. The reason why dropping some signal dimensions for individual sentences does not have an in-

	Sentence 1	Sentence 2	Sentence 3
Drop dim 2 (SD,AU)	i do n t believe it s any, said he. (-0.656)	and we ll just imagine what it would be! (0.945)	at last he came to the top of the mountain, and then he saw something strange. (0.352)
Drop dim 6 (AU)	i do n t want any good. (-0.031)	and i feel as if i could n t say anything about it either. (-0.126)	at last he came to the top of the mountain, and he saw a troop of <unk> <unk>. (0.240)
Drop dim 30 (AU)	i do n t want any good. (0.040)	and i feel as if i could n t say anything about it either. (-0.039)	at last he came to the top of the mountain, and he saw a troop of <unk> <unk>. (0.061)
Drop dim 36 (SD,AU)	did you ever really think so? (-1.477)	they would n t, and i know what it was, and i m afraid. (1.378)	so he took his knapsack and rode off to the palace, and the king and queen were <unk>. (0.614)
Drop dim 48 (SD,AU)	i thought it quite <unk> me. (2.001)	of course, he said, i ca n t have any more, said she. (-0.277)	after that she was a prisoner, and he was very much surprised at all, and was very proud. (-0.870)
Drop dim 49 (SD,AU)	i do n t want any good. (-0.245)	of course, he said; i have n t got anything but myself. (-0.352)	after this he took up the <unk> of his wife, and he went to the <unk> of his <unk>. (-0.643)
Drop dim 51 (SD,AU)	i do hope that s the way, said the old woman. (-1.776)	and she answered: we have all the world that you are, and i will give you a thimble. (-1.179)	at first he came to the castle of the <unk>, and he saw that he was coming. (0.361)
Drop dim 53 (SD,AU)	i do n t want any good. (-0.071)	and i think they re not what i say, and i am not afraid. (0.919)	so the young man was overjoyed to find the king s daughter, and was very fond of him with him. (-1.363)
Drop dim 57 (AU)	i do n t want any good. (-0.037)	and i feel as if i could n t say anything about it either. (-0.080)	at last he came to the top of the mountain, and he saw a troop of <unk> <unk>. (0.192)
Drop dim 59 (SD,AU)	i do n t want any good. (-0.069)	of course, he could n t help thinking of it, but he did n t. (0.512)	and he came back to her, and went to the door, where she could see her. (-1.616)
Drop dim 60 (AU)	i do n t want any good. (-0.068)	and i feel as if i could n t say anything about it either. (-0.053)	at last he came to the top of the mountain, and he saw a troop of <unk> <unk>. (0.076)

Table 5: The results of masking one dimension of mean vector on VAE (C=15, dim=64, CBT, best run). (SD,AU) indicates that this dimension is both a signal dimension and an active unit. (AU) indicates that this dimension is an active unit only.

fluence is that those sentences does not use those dimensions to transmit information. This phenomenon is allowed in the method of signal dimensions because we classify signal and noise dimensions for a number of sentences rather than an individual sentence.

In Table 5, the value of dropped dimensions is also reported between the parenthesis. As the results shown, latent code has relatively small absolute value on some dimensions such that when we mask

one of those dimensions, the reconstructed sentence does not change. The latent codes of sentence 1 on dimension 49, sentence 2 on dimension 48 and 49, and sentence 3 on dimension 2 and 51 have relatively small absolute value, but the results of masking those dimensions show different patterns. Dropping dimension 49 for sentence 1 has the exactly same reconstruction as the mean reconstruction, whereas dropping dimension 48 and 49 for sentence 2 brings nearly completely different reconstructions. Dropping dimension 2 and 51 shows the pattern between them. Compared to the reconstruction obtained by mean, the reconstruction is not completely the same but still shows some similarity especially in the first few words "at last he came to the". This phenomenon suggests that it is not easy to find a unified pattern on signal dimensions for all sentences.

We have demonstrated that the method of signal dimensions is more effective and more accurate on judging whether a dimension in the latent space carries the information from original data. In the next section, we explore signal dimensions with another experiment to probe the information captured by the latent code further.

3.3 Dimension-wise Homotopy

We also do the homotopy evaluation [1] in the signal vector space. The normal homotopy is to sample two latent codes from the standard Gaussian distribution, obtain several intermediate codes between them, pass all codes to the decoder, and generate sentences. Instead of doing a normal homotopy, we leverage a special path and do a dimension-wise homotopy. The recognition of signal and noise dimensions allow us to do this experiment with minimum number of dimensions which carry the information.

We first sample two latent codes as in the normal homotopy and then mask all noise dimensions to obtain two signal vectors. We start from a signal vector and walk along one signal dimension at one time until reach the end. For each dimension, we use a start code, an end code, and three intermediate code to obtain sentences. The end code of the last dimension is the start code of the next dimension. The results are shown in 6. The value on the specific dimension is provided along with the reconstructed sentence to help analysis. This dimension-wise homotopy experiment shows how one sentence can gradually change to another sentence. On dimension 36, because the value of two random sample on this dimension is very close, five sentences are same. On other dimensions, the pattern of neighbour sentences is various. Some of neighbour sentences in five sentences of the same dimension are similar in the first few words. For instance, the second sentence and the third sentence of dimension 53 share the same part "so far as we thought" until comma. Some of neighbour sentences in five sentences of the same dimension are almost completely different. It is hard to say there is any similarity between the fourth sentence of dimension 2, "if a <unk> is not a good, <unk>, i am sure of <unk>.", and the fifth sentence of dimension 2, "give a good hand to her, and let her go, she said." There are also some neighbour sentences which have some similarity but not in the first few words, such as the first sentence of dimension 48, "give a good hand to her, and let her go, she said.", and the second sentence of dimension 48, "that is a <unk> for all, said the story girl, and she is." Both sentences include a common action "said" and the gender of the speaker is the same.

We further notice that the sensitivity of decoder on different dimensions is different. The second sentence and third sentence of dimension 49 have a 0.783 difference on the value, which is larger then the difference of value between the first and fifth sentence of dimension 1, however, they still share the same part "when all was ready", whereas the first sentence and fifth sentence of dimension 1 are almost completely different.

This section has shown that neighbour codes in the latent space can generate sentences with the same first few words or similar structure. In the next section, another experiment is designed to see which aspects of text are extracted and kept by the latent code.

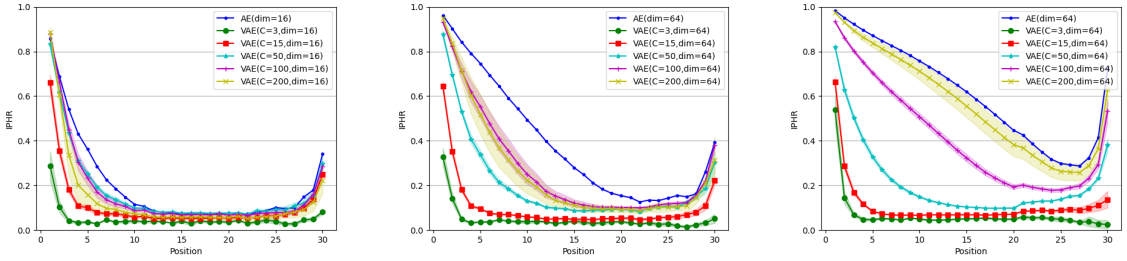
From	if a <unk> would be <unk>, we ll <unk> it, said mrs. rachel.
To	then you are <unk>, you know, he replied.
Dim 2	1. if a <unk> would be <unk>, we ll <unk> it, said mrs. rachel. (0.195) 2. if a <unk> is not to come, mrs. dr. dear, said the story girl. (0.118) 3. if a <unk> is not to come, mrs. dr. dear, <unk> her heart. (0.042) 4. if a <unk> is not a good, <unk>, i am sure of <unk>. (-0.035) 5. give a good hand to her, and let her go, she said. (-0.112)
Dim 36	1. give a good hand to her, and let her go, she said. (0.696) 2. give a good hand to her, and let her go, she said. (0.655) 3. give a good hand to her, and let her go, she said. (0.613) 4. give a good hand to her, and let her go, she said. (0.572) 5. give a good hand to her, and let her go, she said. (0.531)
Dim 48	1. give a good hand to her, and let her go, she said. (-0.165) 2. that is a <unk> for all, said the story girl, and she is. (-0.367) 3. that <unk> the pharisees ; he <unk> his <unk>, and said: o my <unk>. (-0.570) 4. that <unk> the pharisees to say, and <unk>, and never mind. (-0.772) 5. take the path, he cried, and the good lady, with your head! (-0.974)
Dim 49	1. take the path, he cried, and the good lady, with your head! (2.414) 2. when all was ready, the <unk> did not come from her, she said. (1.630) 3. when all was ready, they set out for him, and the king. (0.847) 4. in fact, they were all so busy about that <unk> and not <unk>. (0.063) 5. then he got up to the <unk>, and he did not see her. (-0.720)
Dim 51	1. then he got up to the <unk>, and he did not see her. (0.121) 2. then he remembered what was a <unk>, and he did not know. (-0.139) 3. then he remembered that all the <unk> did not see him. (-0.398) 4. at least, replied the voice, and i was afraid of him. (-0.658) 5. at least, we were all too much for them. (-0.917)
Dim 53	1. at least, we were all too much for them. (-0.482) 2. so far as we thought, he was n t <unk>. (-0.267) 3. so far as we thought, they would be worse. (-0.051) 4. a woman – not a word, but a man s heart. (0.165) 5. a woman – who could bewitch her – so beautiful? (0.380)
Dim 59	1. a woman – who could bewitch her – so beautiful? (0.146) 2. a good woman, he said to her mother s inquiries. (0.658) 3. mrs. elliott, i am sure of all, said she. (1.171) 4. then she said, and how can she come? (1.684) 5. then you are <unk>, you know, he replied. (2.197)

Table 6: Dimension-wise homotopy in signal vector space in VAE (C=15, dim=64, CBT, best run).

3.4 Sentence Chain

We assume that VAEs extract some information from the original data and encode them into latent code. When decoders use the latent code to reconstruct sentences, they can keep the extracted information. Therefore, if we feed the encoder with a sentence and keep feeding the encoder with the new reconstructed sentences, we can obtain a series of sentences which have the information that the encoder of VAEs considers as important for this sentence. We call this series of sentences as a **sentence chain**.

In practice, sentence chain is produced like this: we first feed a sentence to the encoder, and obtain



(a) dim=16, CBT (b) dim=64, CBT (c) dim=64, Wiki

Figure 5: The position correctness of models on CBT and Wiki dataset. The shaded area represents the range of value fluctuating, which is determined by standard deviation.

a decoded sentence; we then feed the decoded sentence to the encoder and obtain a new sentence; this loop stops until we have an invalid sentence from the decoder (i.e., the length of sentence exceeds maximum length or the end of the sentence is not <eos>), or the decoded sentence has appeared before. For sentences which can be perfectly reconstructed by the decoders of VAEs, sentence chains will only contain a single sentence. For sentences which cannot be perfectly reconstructed, sentence chains will have several sentences which should share some same information and should have some similarity. Through sentence chains, we can have a qualitative observation of what kind of information is considered as important by VAEs.

To do this, we randomly sample two sentences from the test set and use them as the start sentences to produce sentence chains. The sentences in sentence chains are shown in Table 7. The first sentence in sentence chain is the start sentence, which in this case is one of the sampled sentences. In sentence chain 1, when we feed the VAE with the start sentence, it seems that the VAE cannot extract any information from this start sentence. The second sentence "and i feel as if i could n t say anything about it either." almost has no similarity with the start sentence. But beside the start sentence, all other sentences in sentence chain 1 have shown some similarity. They all have word "and" as their first word and they all have a pronoun plus a verb after "and". Intuitively those sentences share similar structures which can be indicated by the appearances of "if" and "that", model verb plus "not" or its abbreviation "n t", etc. Similar pattern happens in sentence chain 2. All sentences in sentence chain 2 share the first few words "at last he came to". After the same first part, sentences have "one", "the top" or "the edge" followed by a common "of" and some nouns which represent places. Moreover, all sentences have a comma followed by the second part of sentences. The VAE can extract some information from the start sentence of chain 2 and that kind of information is kept by other sentences in sentence chain. For sentence chain 1, although the information might be not extracted from the start sentence, some kinds of information are still kept. Those kinds of information seem to be the first few words and the structure of the sentence.

3.5 Position Correctness

To demonstrate that VAEs prefer to encode the first few words of a sentence into the latent code, we calculate the Individual Position Hit Rate (IPHR) for models. If the original sentence and reconstructed sentence have the same word at one position, we call it a "hit" at this position. We count the number of hits for every position over all sentences in the test set and divide them with the number of occurrence of positions respectively to obtain IPHR for a model. The IPHRs for different models are depicted in Figure 5. Every model, no matter it is a VAE or an AE, has higher hit rates at the first few positions. This might by caused by the LSTMs in the model. Because LSTM cope with a sentence word by word, it is highly likely that LSTM prefers to memorize the first few words and encode them into the latent code. The end few positions also have slightly higher hit rates. The reason of this behaviour is that the

Sentence chain 1	after all, i d rather love you than not, hurt as it will. and i feel as if i could n t say anything about it either. and she told me that she was n t sorry for me, she said. and i answered: i do not know what to have you, my son. and she told me that she would n t say anything about it at all. and she told him that she did n t know what to make for him. and she told him that she did n t know what to do with it. and he said: i do n t want to know what to do.
Sentence chain 2	at last he came to one of the largest forests in all the world, composed entirely of <unk>. at last he came to the top of the mountain, and he saw a troop of <unk> <unk>. at last he came to the top of the mountain, and the giant was standing there, with his sword. at last he came to the top of the mountain, and the giant was standing there, and he went away. at last he came to the top of the mountain, and the giant was standing there, and he was very angry. at last he came to the edge of the hill, and then he saw that he was sitting by the window. at last he came to the edge of the green forest, and he was as fat as a fish, and he was safe. at last he came to the edge of the green forest, and he was very tired and very glad to see him. at last he came to the edge of the green forest, and he was very busy indeed, and he was very tired. at last he came to the edge of the green forest, and he was as much as ever he could get out. at last he came to the edge of the green forest, and he was sure that he was safe and comfortable. at last he came to the edge of the green forest, and he was sure that he was safe in the water. at last he came to the edge of the green forest, and he was so tired that he was very busy indeed. at last he came to the edge of the green forest, and he was so tired that he could see the smiling pool. at last he came to the edge of the green forest, and he was so tired that he could not get the chance. at last he came to the edge of the green forest, and he had been sure that he was safe in the green forest. at last he came to the edge of the green forest, and he had been very much surprised to see that he was a prisoner. at last he came to the edge of the green forest, and he had been so tired that he could see the tracks of lightfoot.

Table 7: 2 sentence chains in VAE (C=15, dim=64, CBT, best run).

information about the end of a sentence should be encoded into latent code to let the decoder know when to stop decoding.

Above two observations occur not only in VAEs but also in AEs. For VAEs specifically, with the

increase with C value, the hit rate at one position also increases. However, at the first few positions, the hit rate increases more dramatically than other positions. This indicates that higher C value encourages the latent code to encode more information about the first few words. When the C value increases, the IPHR of VAE is approaching the IPHR of AE, but it is hard for VAEs to reach the IPHR of AE.

3.6 Summary

This chapter focused on VAEs for natural text. Five experiments were designed. The first one was using signal dimensions to reconstruct test set to show that signal dimensions carry most information of original data. Dropping one dimension experiment, dimension-wise homotopy experiment, and sentence chain experiment suggest that VAEs are capturing the first few words and the structure of sentences in the latent code for natural text. The position correctness experiment supports the preference of VAEs on first few words.

Natural text is complicated, therefore for better understanding of the behaviour of VAEs, it is required to design datasets which contain a limited set of phenomenon that we could investigate in the latent space. In the next two chapters, two synthetic toy datasets are designed, and experiments and findings on them are reported.

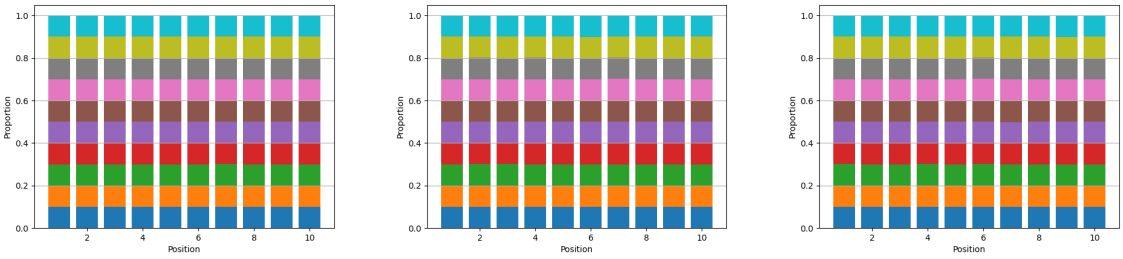
4 Experiments with VAEs on Toy Dataset 1

One behaviour of VAEs found in chapter 3 is that VAEs have the preference to encode the first few words into latent code for natural text. To validate this further, we construct a vocabulary with ten words 'w1', 'w2', 'w3', ..., 'w10' and use the permutations of words in this vocabulary to construct toy dataset 1. We define a rule for sentences in toy dataset 1:

Rule Every word in the vocabulary appears once and only once.

Every sentence which satisfies this rule is a valid sentence in toy dataset 1. Therefore, sentence "w1 w2 w3 w4 w5 w6 w7 w8 w9 w10" is a valid sentence, whereas sentence "w1 w2 w3 w4 w5 w6 w7 w8 w9" and sentence "w1 w1 w3 w4 w5 w6 w7 w8 w9 w10" are invalid. By this setting, we can construct $10! = 3,628,800$ sentences in total and all sentences have a fixed length 10. We split sentences into training, validation and test sets with proportion 60%, 20%, 20%. The size of (training, validation, test) is (2177280, 725760, 725760). To demonstrate that this split does not have bias on words, we plot the frequency of occurrence of words on each position in Figure 6. As the figure shown, every word has the equal frequency on every position. Therefore, toy dataset 1 is a balanced dataset.

We trained VAEs on toy dataset 1. The number of dimensions of word embedding is 8. The hidden state of LSTM is 64 dimensions for both encoder and decoder. The latent space has 4 dimensions. We set C to 2, 4, 8, 12, 16. We trained all models from 3 random starts. We again used Adam [19] with learning rate 0.00075 to optimise objective function. We stopped training when the loss on the validation set did not decrease within 3 epochs, or the difference of loss on the training set within 3 epochs was smaller than 0.001. We set the maximum number of epochs to 50. We also trained a 4-dimension latent space VAE with posterior collapse by setting C to 0 as the baseline. The losses on test set are reported in Table 8. The findings here are similar to chapter 3.



(a) Training set (b) Validation Set (c) Test set

Figure 6: The proportion of words on each position (toy dataset 1). Each colour represents a word.

z-dim=4	Rec.	KL
VAE (collapse)	15.09	0.02
VAE (C=2)	12.95(0.01)	2.2(0.02)
VAE (C=4)	11.00(0.02)	4.19(0.00)
VAE (C=8)	7.40(0.08)	8.07(0.01)
VAE (C=12)	4.53(0.32)	12.03(0.00)
VAE (C=16)	2.02(0.07)	16.03(0.01)

Table 8: Results on the test set. Rec. represents reconstruction loss. For models with several runs, we report the mean and (standard deviation) here.

z-dim=4	OPHR	BLEU-2	Unique %	Rule %
VAE (C=2)	12.57(0.33)	32.56(0.04)	0.22(0.03)	100.00(0.00)
VAE (C=4)	16.10(0.71)	34.42(0.24)	1.73(0.60)	100.00(0.00)
VAE (C=8)	25.48(2.89)	45.21(3.40)	41.91(5.52)	100.00(0.00)
VAE (C=12)	39.51(10.24)	60.04(8.18)	70.42(9.41)	99.12(0.57)
VAE (C=16)	73.94(0.79)	84.83(0.60)	92.03(0.49)	94.65(0.12)

Table 9: Evaluations of reconstruction. For models with several runs, we report the mean and (standard deviation) here.

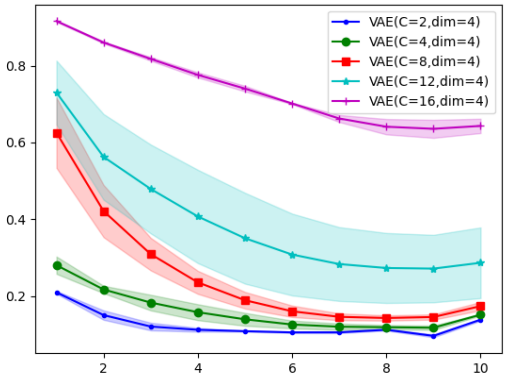


Figure 7: The position correctness of models on toy dataset 1. The shaded area represents the range of value fluctuating, which is determined by standard deviation.

Mean of posterior distributions is used to reconstruct test set. Because sentences in toy dataset 1 consist of isolated words without dependence, to evaluate the reconstruction, for this particular dataset, we calculate Overall Position Hit Rate (OPHR). Recall that if the original sentence and reconstructed sentence have the same word at one position, we call it a "hit" at this position. We count the number of hits over all sentences in the test set and divide them with the total number of occurrence of positions to obtain OPHR for a model. We report OPHR for models in Table 9. BLEU-2 is also reported in the table. Both OPHR and BLEU-2 are increasing with the increase of the C value. Unique rate, which is the number of unique sentences in the reconstruction dividing the number of sentences in the test set, is also reported in Table 9. Unique rate shows that VAEs with higher C value can reconstruct more unique sentences. Higher C value indicates larger amount of information in the latent code. Therefore, it is easier for VAEs to distinguish and reconstruct different sentences. We further calculate the proportion of sentences which satisfy the rule among the unique sentences and report it in the last column. When C exceeds some value, VAEs start to reconstruct some invalid sentences. The decoded sentence is more likely to be an invalid sentence in VAEs with higher C value.

IPHR is also calculated for models of toy dataset 1 and depicted in Figure 7. Results further support that VAEs prefer to encode the first few words. Because compared to natural text, toy dataset 1 does not have bias on position of words in sentences, this experiment on toy dataset 1 is a stronger evidence of the preference on first few words of VAEs.

5 Experiments with VAEs on Toy Dataset 2

Another finding of chapter 3 is that VAEs can encode some information about the structures of sentences into latent code. However, for natural text, the structures of sentences are not explicitly defined which causes the difficulty to observe and understand this behaviour. Therefore, it is required to design sentences with explicitly defined structures to simulate natural text.

Toy dataset 2 is a simulation of natural text. We use Part-of-Speech (POS) to simulate the structure of sentences. We first define the basic structure as "n. v. n. end-punc.", where 'n.' denotes noun, 'v.' denotes verb and 'end-punc.' denotes the punctuation which appears at the end of sentences. Then we define simple sentence structures as "(adj.) n. (adv.) v. (prep.) (adj.) n. end-punc.", where 'adj.' denotes adjective, 'adv.' denotes adverb, 'prep.' denotes preposition and '()' means the part-of-speech can either appear or disappear. Through this way, we have $2^4 = 16$ simple sentence structures, which are shown in Table 10.

We next define complex sentence structure as a combination of two simple sentence structures. We construct complex sentence structures by three methods:

Complex sentence structure 1 conj1. S1 comma S2 end-punc.

Complex sentence structure 2 S1 conj1. S2 end-punc.

Complex sentence structure 3 S1 comma conj2. S2 end-punc.

where 'conj1.' and 'conj2.' denote two kinds of conjunction, 'comma' denotes comma, and 'S1' and 'S2' are two simple sentence structures from which 'end-punc.' is removed. We limit the number of part-of-speech tags that appear in 'S1' and 'S2' to 9 to control the complexity of constructing sentences. Numbers of obtained complex sentence structures are shown in Table 11. Here are three examples of complex sentence structures:

	Structure	No. of Sentences
1	n. v. n. end-punc.	200
2	n. v. adj. n. end-punc.	1,000
3	n. adv. v. n. end-punc.	1,000
4	n. adv. v. adj. n. end-punc.	5,000
5	n. v. prep. n. end-punc.	1,000
6	n. v. prep. adj. n. end-punc.	5,000
7	n. adv. v. prep. n. end-punc.	5,000
8	n. adv. v. prep. adj. n. end-punc.	25,000
9	adj. n. v. n. end-punc.	1,000
10	adj. n. v. adj. n. end-punc.	4,000
11	adj. n. adv. v. n. end-punc.	5,000
12	adj. n. adv. v. adj. n. end-punc.	20,000
13	adj. n. v. prep. n. end-punc.	5,000
14	adj. n. v. prep. adj. n. end-punc.	20,000
15	adj. n. adv. v. prep. n. end-punc.	25,000
16	adj. n. adv. v. prep. adj. n. end-punc.	100,000

Table 10: Simple sentence structures of toy dataset 3.

Length	8	9	10	11	12	Total
Number	1	10	44	112	112	279

Table 11: Number of complex sentence structures of toy dataset 2.

POS (frequency %)	Word
n. (35.73)	dogs cats foxes horses tigers
v. (17.87)	want need have get require
adv. (5.67)	really recently gradually frequently eventually
adj. (11.33)	happy big small beautiful fantastic
prep. (5.67)	on in for to of
conj1. (5.87)	although because when where whereas
conj2. (2.93)	and or
comma (5.87)	,
end-punc. (9.06)	. !

Table 12: Vocabulary of toy dataset 2.

Example 1 conj1. n. v. n. comma n. v. prep. adj. n. end-punc.

Example 2 n. adv. v. adj. n. conj1. n. adv. v. n. end-punc.

Example 3 adj. n. v. n. comma conj2. adj. n. v. prep. n. end-punc.

We choose several words for each part-of-speech to construct sentences. Words are shown in Table 12. For each structure, replacing part-of-speech tags with relevant words, we can construct a huge number of sentences. We limit the number of the appearance of every word in a sentence to one to reduce the complexity of construction. For simple sentence structures, the number of constructed sentences is shown in Table 10. Although this construction does not promise that all sentences are "real" sentences, sentences still can simulate natural text to some degree. The number of constructed sentences can be very large especially for long structures. Therefore, for those structures which can bring more than 10000 sentences, we randomly choose 10000 sentences to build the toy dataset 2. The frequency of the occurrence of each part-of-speech is also reported in Table 12.

We split toy dataset 2 into training, validation and test sets with proportion 60%, 20%, 20%. This proportion is used for every structure to ensure every structure has sentences in the dataset. The final size of (training, validation, test) is (1723680, 574560, 574560). Each dataset does not have any bias on words for the same part-of-speech, as Figure 8 shown. Here are four examples from toy dataset 2:

Example 1 although tigers want big foxes, cats gradually have horses.

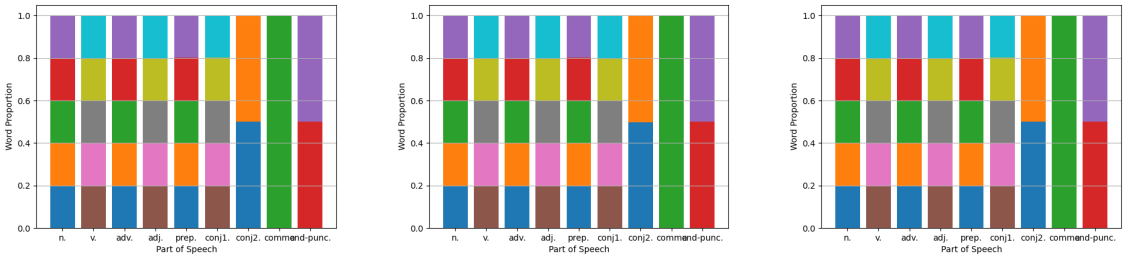
Example 2 tigers need in small cats where dogs want fantastic foxes.

Example 3 tigers require of foxes, and small dogs recently want horses!

Example 4 big tigers really get for foxes.

It can be seen that sentences in toy dataset 2 are somewhat similar to natural text and have some features of natural text although the meaning of sentences is not necessarily meaningful.

We trained VAEs on toy dataset 2. The number of dimensions of word embedding is 16. The hidden state of LSTM is 128 dimensions for both encoder and decoder. The latent space has 16 dimensions.



(a) Training set (b) Validation Set (c) Test set

Figure 8: The frequency of occurrence of words for different part-of-speeches in toy dataset 2. Each colour represents a word.

z-dim=16	Rec.	KL	SD
VAE (collapse)	19.23	0.03	-
VAE (C=4)	15.40(0.10)	4.12(0.02)	2.0(0.0)
VAE (C=8)	12.05(0.34)	8.04(0.02)	3.3(0.9)
VAE (C=16)	5.36(0.33)	16.03(0.01)	5.3(0.5)
VAE (C=32)	0.21(0.05)	31.98(0.01)	9.7(1.7)
VAE (C=64)	0.10(0.08)	63.97(0.02)	16.0(0.0)
Autoencoder	0.50	-	-

Table 13: Results on the test set. Rec. represents reconstruction loss. SD represents the number of signal dimensions. For models with several runs, we report the mean and (standard deviation).

We set C to 4, 8, 16, 32, 64. We trained all models from 3 random starts. We again used Adam [19] with learning rate 0.00075 to optimise objective function. We stopped training when the loss on the validation set did not decrease within 3 epochs, or the difference of loss the training set within 3 epochs was smaller than 0.001. We set the maximum number of epochs to 50. We also trained a 16-dimension latent space VAE with posterior collapse by setting C to 0 as the baseline and a 16-dimension AE. The losses on test set and the number of signal dimensions are reported in Table 13.

The findings here are similar to chapter 3 except that on toy dataset 2, VAEs with high C value such as 32 and 64 can have smaller reconstruction loss than AE. This is probably caused by the explicitly designed structures in toy dataset 2. As shown in Table 1 in chapter 1, compared to AE, VAE can better learn the grammatical information in text. Because the grammatical information is an important part of toy dataset 2, it is reasonable that VAE outperforms AE on this dataset.

5.1 Evaluating the Presence of Structure Information

One key feature of toy dataset 2 is the synthetic structure. Therefore, it is helpful to evaluate the ability of VAEs in capturing this information. Mean of posterior distribution is used to reconstruct test set. The structure of reconstructed sentence is compared with the structure of original sentence. The percentage of structure matches is calculated and reported in Table 14. The results show that higher C value indeed encourages the latent code to learn the structure of sentence. With the increase of C value, more parts of structures of sentences can be reproduced in the reconstruction. We further evaluate the ability of capturing structure information by counting the number of structures produced at the reconstruction phase. The structures used to construct the toy dataset 2 are considered as valid, and everything else as invalid. The numbers of valid and invalid structures are also presented in Table 14. When C value

z-dim=16	Structure Match %	No. of Valid Structure	No. of Invalid Structure
VAE (C=4)	9.41(3.23)	197.33(3.86)	0.00(0.00)
VAE (C=8)	30.53(12.59)	291.00(4.32)	7.00(8.52)
VAE (C=16)	67.36(4.90)	295.00(0.00)	704.00(262.84)
VAE (C=32)	98.86(0.69)	295.00(0.00)	753.67(196.55)
VAE (C=64)	99.29(0.84)	295.00(0.00)	396.33(366.65)

Table 14: Evaluations of capturing structure information on toy dataset 2. We report the mean and (standard deviation) here.

z-dim=16	n. %	v. %	adv. %	adj. %
Baseline	20.00	20.00	20.00	20.00
VAE(C=4)	45.73(6.79)	19.92(0.20)	20.28(0.33)	19.87(0.16)
VAE(C=8)	83.48(8.21)	19.96(0.06)	19.97(0.13)	20.04(0.06)
VAE(C=16)	90.77(1.63)	90.42(3.62)	87.66(2.94)	19.97(0.04)
VAE(C=32)	99.71(0.15)	99.78(0.21)	99.66(0.19)	99.72(0.19)
VAE(C=64)	99.83(0.15)	99.81(0.20)	99.67(0.35)	99.77(0.26)
z-dim=16	prep. %	conj1. %	conj2. %	end-punc. %
Baseline	20.00	20.00	50.00	50.00
VAE(C=4)	19.93(0.19)	19.85(0.41)	62.61(17.94)	49.88(0.10)
VAE(C=8)	19.88(0.11)	20.11(0.35)	81.27(21.99)	87.25(9.28)
VAE(C=16)	83.08(6.51)	36.55(23.20)	96.85(3.16)	91.96(2.80)
VAE(C=32)	99.66(0.25)	99.93(0.00)	99.96(0.01)	99.78(0.17)
VAE(C=64)	99.79(0.24)	99.90(0.12)	99.99(0.02)	99.88(0.11)

Table 15: The correctness of words for each part-of-speech. We report the mean and (standard deviation) here.

is relatively small such as C=4, VAEs cannot learn all valid structures, whereas when C reaches some value, all valid sentences can be learned. The reason of this might be that the constraint on encoded information, which is indicated by C value, does not allow the latent code to encode all structures. When C value is relatively large ($C \geq 16$), the reconstruction can produce many invalid structures. However, the number of sentences with invalid structures is still small, as indicated by the percentage of structure match.

5.2 Evaluating the Presence of Part-of-Speech Information

Part-of-speech is the component of sentence structure. In toy dataset 2, several words are chosen for each of part-of-speech. The evaluation of how well VAEs encode the right word for right part-of-speech is helpful to understand the behaviour of VAEs. To do this, we count the number of correct words for each part-of-speech and divide it to the number of appearance of each part-of-speech. This calculation is only done on the sentences whose structures have been correctly reproduced in the mean reconstruction of test set. The results are provided in Table 15. The baseline of this figure can be obtained from Figure 8. With the increase of C value, the correctness on all part-of-speeches gains improvement. When $C = 4$, only 'n.' and 'conj2.' have a better correctness than the baseline. When C increases to 8, the correctness on 'end-punc.' also improves. When $C = 16$, all part-of-speeches except 'adj.' have improvements and when C reaches 32, all part-of-speeches have a significant increase compared to the baseline. Ranking this improvement based on part-of-speech, the order is 'n.', 'conj2.', 'end-punc.', 'v.', 'adv.', 'prep.',

sentence	happy horses need on big foxes although cats want dogs!	happy horses recently need foxes.	cats require fantastic dogs, or foxes want in horses.
Mean	beautiful horses need on happy foxes because cats need tigers.	beautiful horses recently need foxes.	cats require happy dogs, or foxes want in horses.
Active units	beautiful horses need on happy foxes because cats need tigers.	beautiful horses recently need foxes.	cats require happy dogs, or foxes want in horses.
Signal Vector	beautiful horses need on happy foxes because cats need tigers.	beautiful horses recently need foxes.	cats require happy dogs, or foxes want in horses.
Drop dim 4 (SD,AU)	beautiful horses need on happy dogs whereas cats get foxes! (0.120)	happy horses recently need dogs, and tigers require to cats. (2.252)	tigers get cats, and foxes get of horses! (-0.350)
Drop dim 5 (SD,AU)	beautiful horses need on happy foxes because cats need tigers. (0.011)	beautiful horses recently need foxes. (0.057)	cats have horses, and tigers get for foxes! (0.848)
Drop dim 6 (SD,AU)	whereas beautiful cats need of small horses, dogs get tigers. (-1.186)	beautiful cats get dogs where foxes eventually require horses. (-1.509)	cats require in dogs, or foxes want in horses! (0.375)
Drop dim 7 (SD,AU)	beautiful horses need on happy foxes, and cats want dogs! (-0.881)	beautiful horses recently need foxes. (0.042)	cats require happy dogs, or foxes want in horses. (0.867)
Drop dim 14 (SD,AU)	cats frequently want happy foxes because horses have for tigers. (-2.286)	beautiful cats eventually need foxes. (-0.619)	horses require big foxes, and tigers want big cats. (0.473)
Drop dim 16 (SD,AU)	happy horses get to tigers whereas cats gradually have dogs! (0.596)	happy dogs gradually want happy tigers! (-2.522)	tigers get cats, and horses have to big dogs. (-0.173)

Table 16: The results of masking one dimension of mean vector on VAE (C=16, best run). (SD,AU) indicates that this dimension is both a signal dimension and an active unit. (AU) indicates that this dimension is an active unit only.

'conj1.' and 'adj.'. The reason for this order is not intuitive but this behaviour of preference on different part-of-speeches indeed happens. We leave further exploration of this to future work.

5.3 Masking One Dimension and Dimension-wise Homotopy

The experiments on masking one dimension, and dimension-wise homotopy in chapter 3 are reported for toy dataset 2 to obtain qualitative observation of what has been captured by VAEs. The results of masking experiment are presented in Table 16. Using mean, active units and signal vector have the same reconstruction, similar to the results in chapter 3. Although mean reconstruction is not completely same as the original sentence, the structure of reconstruction is identical to the structure of original sentence in these three cases. In most cases, dropping one dimension does not change the structure of sentence rapidly. Nevertheless, there is no clear correlation between the dropped value and structure changes. Masking dimension with larger absolute value does not mean that the structure of sentence changes

From	dogs eventually require in horses whereas cats get foxes!
To	although tigers require for dogs, foxes frequently want cats!
Dim 4	1. dogs eventually require in horses whereas cats get foxes! (0.092) 2. dogs eventually require in horses whereas cats get foxes. (0.281) 3. dogs eventually require in happy tigers because horses have foxes. (0.470) 4. dogs eventually require in happy tigers when cats want foxes. (0.660) 5. dogs eventually require in happy tigers whereas cats need foxes. (0.849)
Dim 5	1. dogs eventually require in happy tigers whereas cats need foxes. (-0.178) 2. dogs eventually require in happy tigers whereas cats need foxes. (-0.103) 3. dogs eventually require in happy tigers whereas cats need foxes. (-0.028) 4. dogs eventually require in happy tigers whereas cats need foxes. (0.047) 5. dogs eventually require in happy tigers whereas cats need foxes. (0.122)
Dim 6	1. dogs eventually require in happy tigers whereas cats need foxes. (-0.919) 2. dogs eventually require happy horses whereas cats have tigers. (-0.356) 3. dogs eventually require happy horses whereas tigers get foxes. (0.208) 4. dogs require require happy cats although horses gradually require foxes. (0.771) 5. dogs require require horses small cats want foxes eventually. (1.335)
Dim 7	1. dogs require require horses small cats want foxes eventually. (-2.078) 2. dogs require require horses small cats want foxes eventually. (-1.403) 3. dogs require require horses small cats want foxes eventually. (-0.729) 4. dogs require require horses small cats want foxes. (-0.055) 5. dogs require require horses small cats want foxes. (0.620)
Dim 14	1. dogs require require horses small cats want foxes. (2.220) 2. dogs require require big horses, and cats need foxes. (1.877) 3. dogs require horses, or beautiful cats eventually require foxes. (1.535) 4. dogs require happy cats, or horses have foxes. (1.192) 5. dogs require happy horses, or tigers want foxes! (0.850)
Dim 16	1. dogs require happy horses, or tigers want foxes! (2.044) 2. dogs want happy tigers, and horses require cats! (1.207) 3. although dogs have for beautiful foxes, tigers require big cats. (0.371) 4. although foxes require of tigers, dogs gradually need horses. (-0.466) 5. although tigers require for dogs, foxes frequently want cats! (-1.303)

Table 17: Dimension-wise homotopy in signal vector space in VAE (C=16, best run).

more. Dropping dimension 6 for sentence 2 changes the structure of sentence from "adj. n. adv. v. n. end-punc." to "adj. n. v. n. conj1. n. adv. v. n. end-punc.", from a simple sentence structure to a complex sentence structure, whereas dropping dimension 16 only changes it to "adj. n. adv. v. adj. n. end-punc.", from a simple sentence structure to another simple sentence structure. However, the absolute value of dropped dimension 16 is larger than that of dimension 6. Unlike the results of natural text, the latent code does not necessarily encode the first word. This is because the structure of sentence is a more important feature of toy dataset 2 and only first few words cannot represent the structure.

The results of dimension-wise homotopy for same VAE are shown in Table 17. In this case, the intermediate sentences can be invalid. For example, 5 sentences of dimension 7 are all invalid because they repeat 'v.' ('require') once. The reason of this might be that the latent code of those sentences is located in the region where the decoder has not observed during training and does not know how to encode. Because two initial latent codes are sampled from prior distribution, it is worth mentioning that this observation is subject to the random sampling.

Through this experiment, we can observe how the structure of sentence gradually changes from one

Group 1	Structure 1: conj1. n. adv. v. prep. adj. n. comma n. v. n. end-punc. Structure 2: adj. n. v. n. comma conj2. n. v. prep. n. end-punc. Structure 3: n. v. n. conj1. n. adv. v. prep. n. end-punc.
Group 2	Structure 1: conj1. n. v. adj. n. comma adj. n. v. n. end-punc. Structure 2: conj1. adj. n. adv. v. prep. n. comma n. v. n. end-punc. Structure 3: n. v. n. conj1. adj. n. adv. v. n. end-punc.
Group 3	Structure 1: conj1. n. v. n. comma adj. n. v. adj. n. end-punc. Structure 2: n. v. adj. n. end-punc. Structure 3: conj1. n. v. n. comma adj. n. adv. v. prep. n. end-punc.

Table 18: Three groups of sentence structures.

structure to another structure. Examples of changes of the structure of sentence in changing from "n. adv. v. prep. n. conj1. n. v. n. end-punc." to "conj1. n. v. prep. n. comma n. adv. v. n. end-punc." are:

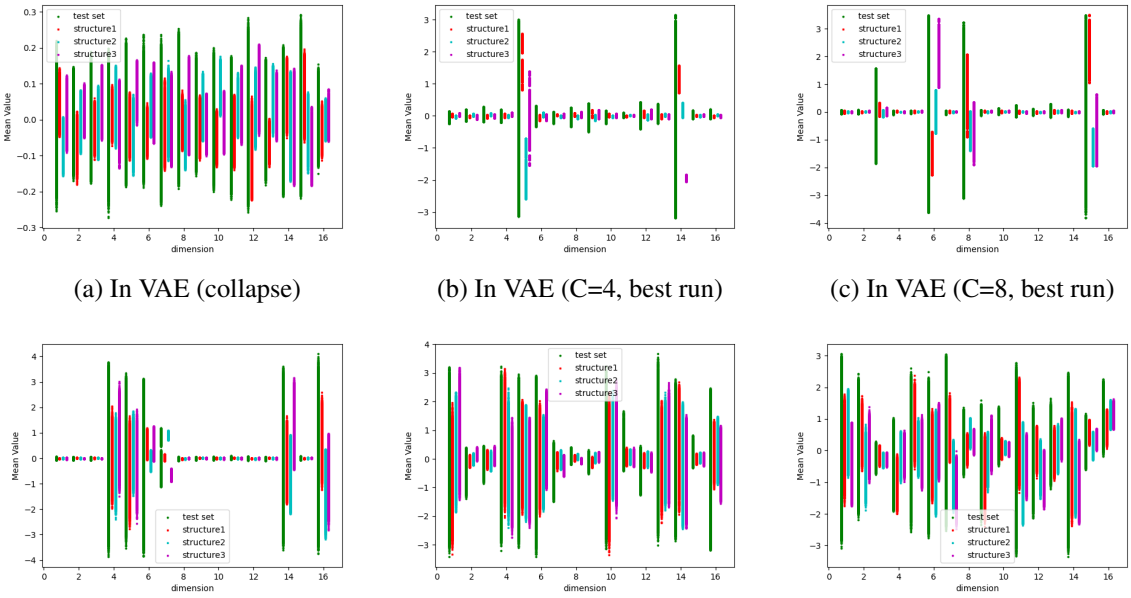
1. from "n. adv. v. prep. n. conj1. n. v. n. end-punc." to "n. adv. v. prep. adj. n. conj1. n. v. n. end-punc." in dimension 4, adding an 'adj.';
2. from "n. v. v. n. adj. n. v. n. adv. end-punc." to "n. v. v. n. adj. n. v. n. end-punc." in dimension 7, removing an 'adv.';
3. from "n. v. v. n. adj. n. v. n. end-punc." to to "n. v. n. comma conj2. adj. n. adv. v. n. end-punc." to "n. v. adj. n. comma conj2. n. v. n. end-punc." in dimension 14, the first transformation is from an invalid structure to an valid structure, and the second transformation is adding an 'adj.' in the first clause and removing an 'adj.' and an 'adv.' in the second clause;
4. from "n. v. adj. n. comma conj2. n. v. n. end-punc." to "conj1. n. v. prep. adj. n. comma n. v. adj. n. end-punc." to "conj1. n. v. prep. n. comma n. adv. v. n. end-punc.", the first transformation is adding a 'prep.' to the first clause, adding an 'adj.' to the second clause and connecting two clauses from using 'conj2.' to using 'conj1.' and the second transformation is removing 'adj.' in two clauses and adding an 'adv.' to the second clause.

Structural information is an important characteristic of toy dataset 2. This section has shown how the changes of the latent code influence the structural information in the relevant reconstructed sentences. The next section focuses on how VAEs differentiate sentence structures.

5.4 Using Latent Code to Distinguish Different Sentence Structures

Previous results show that VAEs with large C value can correctly reproduce the structure of original sentence. One question about the behaviour of VAEs on toy dataset 2 is how the decoder of VAE distinguishes different sentence structures. The latent code might have some characteristics to help doing this. To validate this assumption, we sample three groups of different structures from sentence structures which were used to construct toy dataset 2. Each group has three sentence structures. The structures are presented in Table 18. Using these structures, we can use previous method to construct sentences rather than using sentences in toy dataset 2. This time, for those structures, we randomly choose 20000 sentences because the aim is to know how VAEs distinguish different structures, which should be independent from the number of sentences for each structure.

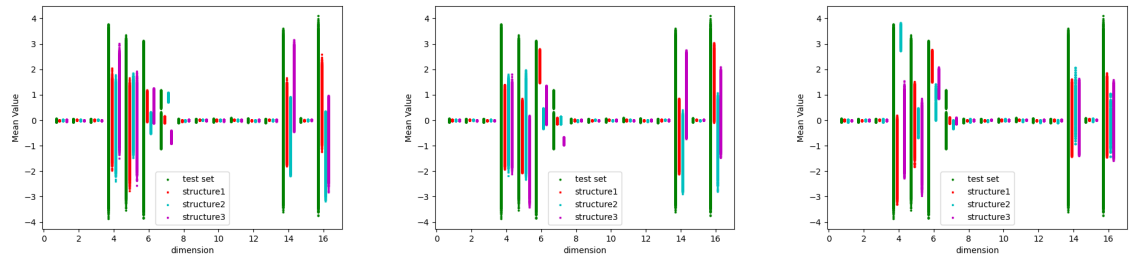
We first use structures in group 1 to construct new sentences. The mean of posterior distribution is used as the latent code for a sentence. For one VAE, we obtain the value of latent code on each



(a) In VAE (collapse) (b) In VAE (C=4, best run) (c) In VAE (C=8, best run)

(d) In VAE (C=16, best run) (e) In VAE (C=32, best run) (f) In VAE (C=64, best run)

Figure 9: The value of latent code of new constructed sentences and sentences in test set. Structures are from Group 1.



(a) Group 1 (b) Group 2 (c) Group 3

Figure 10: The value of latent code of sentences in test set and sentences constructed by structures of three groups, in VAE (C=16, best run).

dimension for these new sentences and sentences in test set and plot them. The figures for different VAEs are shown together in Figure 9. Points are labeled and coloured based on where sentences belong to.

In the posterior collapse situation, Figure 9a, there is no significant difference between structures. When C equals to 4, on dimension 14, there is a clear and intuitive difference between three structures. It is easy to use different ranges on this dimension to distinguish these three structures. Same situation also happens on dimension 6 of VAE (C=8, best run) and dimension 7 of VAE (C=16, best run). This result indicates that VAEs can use different ranges on some dimensions to recognize different sentence structures. However, in VAE (C=32, best run) and VAE (C=64, best run), there is no such pattern. The reason might be that high C value allows VAEs to utilize different regions or dimensions of latent space.

The second experiment leverages all structures in three groups to construct new sentences. Similar to last experiment, we obtain the value of latent code on each dimension for these new sentences and sentences in test set and plot them in Figure 10. The three structures of group 1 can be distinguished by different ranges on dimension 7. The three structures of group 2 cannot be distinguished directly,

however, structure 3 of group 2 can be easily recognized by dimension 7 and the distinction between structure 1 and structure 2 in group 2 is clear on dimension 6. As for three structures in group 3, there is no intuitive difference between structure 1 and structure 3 on the value of latent code. Only structure 2 of group 3 can be simply separated through dimension 4. It is not guaranteed that all structures can be recognized and distinguished through this method, however, for some structures, the difference is clear and distinguishable on the latent code of VAEs.

5.5 Summary

In this chapter, a synthetic dataset was designed to help with understanding how VAEs capture the structure information in sentences. To this end, we first defined simple sentence structures and then used them to further design complex sentence structures. Some words were used along with structures to construct sentences. Several experiments were done on VAEs of this dataset. The first experiment focused on the structure information captured by VAEs. The second experiment focused on part-of-speech tags. Results show that with the increase of C value, both structure information and word information have been captured better. Masking and the dimension-wise homotopy experiments were also done in this chapter to observe how the changes of the latent code influence the structure of sentence. The last experiment focused on the latent code for different structures. Results indicate that latent code can be leveraged to distinguish sentence structures.

6 Conclusion and Future Work

In this dissertation, we have done several experiments to understand the latent space in VAEs. We first introduced the concept of signal dimensions in chapter 2 to reduce the dimensionality of the latent space in the following experiments. Then we evaluated VAEs on natural text by highlighting the importance of signal dimensions, masking experiment, dimension-wise homotopy experiment, sentence chain experiment and position correctness experiment in chapter 3. We found that:

- for natural text, VAEs have the preference to encode the first few words into the latent code;
- for natural text, VAEs can capture some information about the structure of sentence in the latent code.

To validate and further support these two findings, two synthetic toy datasets were designed. Toy dataset 1 was used to validate and support the first finding. In chapter 4, the details of the implementation of toy dataset 1 were introduced and several experiments were implemented. Results again indicated the preference of VAEs on first few words. Toy dataset 2 was used to support the second finding. In chapter 5, the details about how to construct structured sentences in toy dataset 2 were presented. Several experiments were implemented on VAEs of toy dataset 2. Results supported the conclusion that VAEs can potentially capture structure information.

Several questions arises from this work. Can the structure information of sentences captured by the latent code of VAEs be expressed explicitly? Is the preference of VAEs on the first few words due to LSTMs used in VAEs? Will different choices of the encoder and decoder in VAEs influence the captured information in the latent code? We hope future works could focus on explaining these problems, and build on the findings of our work.

References

[1] Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Józefowicz, and Samy Bengio, "Generating sentences from a continuous space," in *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016*, Berlin, Germany, August 2016, pp. 10–21, ACL.

[2] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[3] Diederik P. Kingma and Max Welling, "Auto-encoding variational bayes," in *2nd International Conference on Learning Representations, ICLR 2014, Conference Track Proceedings*, Banff, AB, Canada, April 2014.

[4] Otto Fabius, Joost R. van Amersfoort, and Diederik P. Kingma, "Variational recurrent auto-encoders," in *3rd International Conference on Learning Representations, ICLR 2015, Workshop Track Proceedings*, San Diego, CA, USA, May 2015.

[5] Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C. Courville, and Yoshua Bengio, "A recurrent latent variable model for sequential data," in *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015*, Montreal, Quebec, Canada, December 2015, pp. 2980–2988.

[6] Matt J. Kusner, Brooks Paige, and José Miguel Hernández-Lobato, "Grammar variational autoencoder," in *Proceedings of the 34th International Conference on Machine Learning, ICML 2017*, Sydney, NSW, Australia, August 2017, vol. 70 of *Proceedings of Machine Learning Research*, pp. 1945–1954, PMLR.

[7] Christian Robert and George Casella, *Monte Carlo statistical methods*, Springer Science & Business Media, 2013.

[8] Zichao Yang, Zhiting Hu, Ruslan Salakhutdinov, and Taylor Berg-Kirkpatrick, "Improved variational autoencoders for text modeling using dilated convolutions," in *Proceedings of the 34th International Conference on Machine Learning, ICML 2017*, Sydney, NSW, Australia, August 2017, vol. 70 of *Proceedings of Machine Learning Research*, pp. 3881–3890, PMLR.

[9] Adji B. Dieng, Yoon Kim, Alexander M. Rush, and David M. Blei, "Avoiding latent variable collapse with generative skip models," in *The 22nd International Conference on Artificial Intelligence and Statistics, AISTATS 2019*, Naha, Okinawa, Japan, April 2019, vol. 89 of *Proceedings of Machine Learning Research*, pp. 2397–2405, PMLR.

[10] Junxian He, Daniel Spokoyny, Graham Neubig, and Taylor Berg-Kirkpatrick, "Lagging inference networks and posterior collapse in variational autoencoders," in *7th International Conference on Learning Representations, ICLR 2019*, New Orleans, LA, USA, May 2019.

[11] Ali Razavi, Aäron van den Oord, Ben Poole, and Oriol Vinyals, "Preventing posterior collapse with delta-vaes," in *7th International Conference on Learning Representations, ICLR 2019*, New Orleans, LA, USA, May 2019.

[12] Irina Higgins, Loïc Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner, "beta-vae: Learning basic visual concepts with a constrained variational framework," in *5th International Conference on Learning Representations, ICLR 2017, Conference Track Proceedings*, Toulon, France, April 2017.

[13] Christopher P. Burgess, Irina Higgins, Arka Pal, Loïc Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner, "Understanding disentangling in β-vae," *CoRR*, vol. abs/1804.03599, 2018.

[14] Stephen Boyd and Lieven Vandenberghe, *Convex optimization*, Cambridge university press, 2004.

[15] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[16] Felix A. Gers, Jürgen Schmidhuber, and Fred Cummins, "Learning to forget: Continual prediction with lstm," *Neural Computation*, vol. 12, no. 10, pp. 2451–2471, 2000.

[17] Yuri Burda, Roger B. Grosse, and Ruslan Salakhutdinov, "Importance weighted autoencoders," in *4th International Conference on Learning Representations, ICLR 2016, Conference Track Proceedings*, San Juan, Puerto Rico, May 2016.

[18] Victor Prokhorov, Ehsan Shareghi, Yingzhen Li, Mohammad Taher Pilehvar, and Nigel Collier, "On the importance of the kullback-leibler divergence term in variational autoencoders for text generation," in *Proceedings of the 3rd Workshop on Neural Generation and Translation@EMNLP-IJCNLP 2019*, Hong Kong, November 2019, pp. 118–127, Association for Computational Linguistics.

[19] Diederik P. Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, Conference Track Proceedings*, San Diego, CA, USA, May 2015.

[20] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, Philadelphia, Pennsylvania, USA, July 2002, pp. 311–318, Association for Computational Linguistics.

Appendix A Proofs

A.1 Proof of the Closed-form Expression of KL Divergence Term

Assume that random variables $X_1, X_2, ..., X_n$ are mutual independent. P, Q are two multivariate probability density functions of these random variables. We have:

$$P(x_1, x_2, ..., x_n) = p(x_1)p(x_2)\dots p(x_n) \tag{A.1}$$
$$Q(x_1, x_2, ..., x_n) = q(x_1)q(x_2)\dots q(x_n) \tag{A.2}$$

Then,

$$\begin{aligned}
\mathrm{KL(P||Q)} &= \int \cdots \int P(x_1, x_2, ..., x_n) \log \frac{P(x_1, x_2, ..., x_n)}{Q(x_1, x_2, ..., x_n)}\, dx_1 \dots dx_n \\
&= \int \cdots \int p(x_1)p(x_2)\dots p(x_n) \log \frac{p(x_1)p(x_2)\dots p(x_n)}{q(x_1)q(x_2)\dots q(x_n)}\, dx_1 \dots dx_n \\
&= \int \cdots \int p(x_1)p(x_2)\dots p(x_n) \log \frac{p(x_1)}{q(x_1)}\, dx_1 \dots dx_n \\
&\quad + \cdots + \int \cdots \int p(x_1)p(x_2)\dots p(x_n) \log \frac{p(x_n)}{q(x_n)}\, dx_1 \dots dx_n \\
&= \int p(x_1) \log \frac{p(x_1)}{q(x_1)}\, dx_1 + \cdots + \int p(x_n) \log \frac{p(x_n)}{q(x_n)} dx_n \\
&= \sum_{i=1}^{n} \int p(x_i) \log \frac{p(x_i)}{q(x_i)}\, dx_i
\end{aligned} \tag{A.3}$$

When P is a multivariate diagonal Gaussian distribution and Q is a standard Gaussian distribution, we have $p(x_i) \sim \mathcal{N}(\mu_i, \sigma_i^2)$, $q(x_i) \sim \mathcal{N}(0, 1)$ and

$$\begin{aligned}
\int p(x_i) \log \frac{p(x_i)}{q(x_i)}\, dx_i &= \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}\sigma_i} e^{-\frac{(x_i-\mu_i)^2}{2\sigma_i^2}} \log \frac{\frac{1}{\sqrt{2\pi}\sigma_i} e^{-\frac{(x_i-\mu_i)^2}{2\sigma_i^2}}}{\frac{1}{\sqrt{2\pi}} e^{-\frac{x_i^2}{2}}}\, dx_i \\
&= -\log \sigma_i - \int_{-\infty}^{\infty} \frac{(x_i-\mu_i)^2}{2\sqrt{2\pi}\sigma_i^3} e^{-\frac{(x_i-\mu_i)^2}{2\sigma_i^2}}\, dx_i \\
&\quad + \int_{-\infty}^{\infty} \frac{x_i^2}{2\sqrt{2\pi}\sigma_i} e^{-\frac{(x_i-\mu_i)^2}{2\sigma_i^2}}\, dx_i \\
&= -\log \sigma_i - \int_{-\infty}^{\infty} \frac{y_i^2}{\sqrt{\pi}} e^{-y_i^2}\, dy_i + \int_{-\infty}^{\infty} \frac{(\sqrt{2}\sigma_i y_i + \mu_i)^2}{2\sqrt{\pi}} e^{-y_i^2}\, dy_i \\
&= -\log \sigma_i + \frac{\sigma_i^2 - 1}{\sqrt{\pi}} \int_{-\infty}^{\infty} y_i^2 e^{-y_i^2}\, dy_i + \frac{\mu_i^2}{2\sqrt{\pi}} \int_{-\infty}^{\infty} e^{-y_i^2}\, dy_i \\
&= -\log \sigma_i + \frac{\sigma_i^2 - 1}{2} + \frac{\mu_i^2}{2} \\
&= \frac{\mu_i^2 + \sigma_i^2 - 1 - \log \sigma_i^2}{2}
\end{aligned} \tag{A.4}$$

Hence

$$\mathrm{KL(P||Q)} = \frac{1}{2} \sum_{\mathrm{i}=1}^{\mathrm{n}} (\mu_{\mathrm{i}}^2 + \sigma_{\mathrm{i}}^2 - 1 - \log \sigma_{\mathrm{i}}^2) \tag{A.5}$$