

國立臺灣大學電機資訊學院資訊工程學系

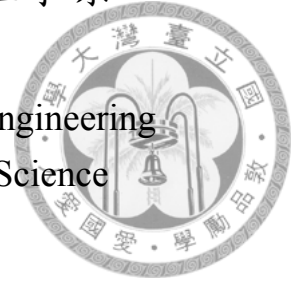
碩士論文

Department of Computer Science and Information Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Master Thesis



條件式句子改寫且不使用成對資料訓練

Conditional Sentence Rephrasing without Pairwise  
Training Corpus

李彥霆

Yen-Ting Lee

指導教授：林守德博士

Advisor: Shou-De Lin, Ph.D.

中華民國 108 年 7 月

July, 2019



## 誌謝

首先，我想感謝指導教授林守德教授，在做研究的過程中給了我很多的指導與建議，讓我可以順利的完成碩士論文。接著，我想感謝所有口試委員們。最後我想感謝沈亮欣、戴培倫、林祐萱三位在實驗室同學們，在研究改寫模型的路上一起尋找論文和實驗。



# Acknowledgements

First, I would like to express my gratitude to my advisor, Prof. Shou-De Lin, who give me many advises to improve my research work and finish this thesis. Second, I would like to thank all the defense committee members. Last but not the least, I would like to thank Liang-Hsin Shen, Pei-Lun Tai, and Amy Lin for surveying and experimenting in the field of rephrasing together.



## 摘要

自然語言生成在最近發展的相當蓬勃，無論是基於對抗式生成網路 (GAN) 或是變分自動編碼器 (VAE)，都有相當的佳作發表。在自然語言生成的領域中，條件式的改寫是較少人專注的題目，在這篇論文中，我們對這個題目有更正式的定義—將句子依據給定的條件改寫，且生成的句子需和原句相像並滿足給定的條件。我們提出了一個基於序列變分自動編碼器的模型解決這個問題。這個模型在訓練時和自動編碼器相同，輸入和目標是同個句子，但我們額外加入了條件提醒的機制，讓模型在生成句子時會去注意在我們給定的條件上，達成控制的目標。最後我們的實驗結果支持這個模型能生成好品質的句子並符合條件的改寫。

關鍵字：自然語言生成, 非監督式機器學習



# Abstract

Natural language generation has been a popular field with lots of quality works published based on generative adversarial network (GAN) or variational autoencoder (VAE). However, rephrasing with condition is a problem that few people focus on. In this work, the problem is formally defined as "rephrase a sentence with given condition, and the generated sentence should be similar to the origin sentence and it should satisfy the given condition". Moreover, we propose a conditional model based on sentence-VAE to solve the problem. The model is trained as an autoencoder, but we can control the condition of the generated sentence. And, it inherits the nature of autoencoder that the generated sentences would be similar to the input sentence. With experiment results supported, the model can solve the problem with quality sentences.

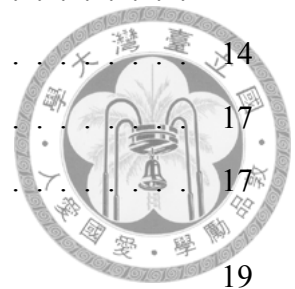
Keywords: natural language generation, variational autoencoder, unsupervised machine learning



# Contents

|  |     |
|--|-----|
| 誌謝   | i   |
| Acknowledgements   | ii  |
| 摘要   | iii |
| Abstract   | iv  |
| 1 Introduction   | 1   |
| 2 Related Works  | 3   |
| 2.1 Sentence-VAE . . . . .                                 | 3   |
| 2.2 Prototype Editing . . . . .                            | 4   |
| 3 Problem Definition                                       | 5   |
| 3.1 Definition . . . . .                                   | 5   |
| 3.2 Example . . . . .                                      | 6   |
| 4 Proposed Method  | 7   |
| 4.1 Conditional Sentence Variational Autoencoder . . . . . | 7   |
| 4.2 Condition Mechanism . . . . .                          | 9   |
| 5 Experiments  | 10  |
| 5.1 Dataset . . . . .                                      | 10  |
| 5.2 Condition Evaluation . . . . .                         | 11  |
| 5.2.1 Condition Function . . . . .                         | 11  |

|   |    |
|---|----|
| 5.2.2 Accuracy . . . . .                    | 12 |
| 5.2.3 Generated Sentences . . . . .         | 14 |
| 5.3 Generation Quality . . . . .            | 17 |
| 5.4 Can An Autoencoder Also Work? . . . . . | 17 |
| 6 Conclusions and Future Work               | 19 |
| Bibliography                                | 20 |





# List of Figures

|     |   |    |
|-----|---|----|
| 4.1 | A Condition Variational Autoencoder Overview. . . . . | 7  |
| 4.2 | Training as an autoencoder. . . . .                   | 8  |
| 4.3 | Training as an autoencoder. . . . .                   | 8  |
| 5.1 | Length Distribution. . . . .                          | 12 |
| 5.2 | Length Accuracy. . . . .                              | 13 |
| 5.3 | Autoencoder Accuracy. . . . .                         | 18 |





# List of Tables

|      |  |    |
|------|--|----|
| 3.1  | An example about length condition rephrasing . . . . . | 6  |
| 5.1  | Number of sentences in each set . . . . .              | 10 |
| 5.2  | Tense Condition Accuracy . . . . .                     | 14 |
| 5.3  | Subject Condition Accuracy . . . . .                   | 14 |
| 5.4  | Subject Condition Accuracy . . . . .                   | 14 |
| 5.5  | Length Examples . . . . .                              | 15 |
| 5.6  | Tense Examples . . . . .                               | 15 |
| 5.7  | Subject Examples . . . . .                             | 16 |
| 5.8  | Single or Plural Examples . . . . .                    | 16 |
| 5.9  | Copy Rate . . . . .                                    | 17 |
| 5.10 | Bigram and Trigram Probability . . . . .               | 18 |
| 5.11 | Jaccard Distance . . . . .                             | 18 |



# Chapter 1

## Introduction

Natural Language Generation (NLG) has been thriving in these years as generative adversarial networks (GAN) [6] and variational autoencoder (VAE) [10][13] have been applied to sequence-to-sequence [16] models. Besides sequence-GAN [18] and sentence-VAE [4], works in machine translation also improve the quality of generation using attention mechanism [2][12] and even a new model of sequence-to-sequence based on attention mechanism [17].

Many works have worked on generating with styling [5], editing [7] and generating from continuous space [15] [4]. But we have not seen works to rephrase sentence with condition, which cannot be achieved by any of exists model if parallel corpus cannot be used. Conditions may vary, and parallel corpus seldom exists. This is a problem worth solving.

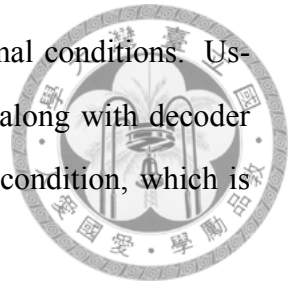
With the understanding of how a recurrent neuron network works [9][11], a condition may be controlled by several cells. For example, an encoder of an autoencoder may capture the length information in some cells, so that the decoder knows how many words to decode. Based on this discovery, we may ask what if we give the condition explicitly?

Holding this idea, we propose a conditional variational autoencoder (CS-VAE) to solve the conditional rephrasing problem. Combining continuous generating and condition mechanism, we expect the model to learn the condition information using our explicitly given feature, so that the latent space can unsupervised cluster similar sentences with different conditions. If the model is well trained, then we can manipulate the condition to

control the output sentence.

We train the model like a variational autoencoder with additional conditions. Using KL annealing trick [4], our model learns posterior distribution along with decoder language model. At testing phase, we input both a sentence and a condition, which is different from traditional variational autoencoder scheme.

The model is experimented with several condition functions including length, subject, tense and "single or plural". The results show that our model does have the ability to rephrase sentences with given condition, and our model has low copy rate which means it often generates sentences not in the training set. The smoothness is also examined by bi-gram and tri-gram language model, which represent part of the generation quality.





## Chapter 2

### Related Works

#### 2.1 Sentence-VAE

Sentence VAE [4] is a generative sequence-to-sequence model based on variational autoencoder [10]. The sequence autoencoder is modified by adding a re-sample of the encoder hidden state. Instead of a deterministic encoder, the S-VAE encoder describe a posterior distribution of the latent space  $q(\vec{z}|x)$ , and then decoder is a language model from the latent space, which modeling  $p(x|\vec{z})$ . The latent space is generally a standard normal distribution  $N(0, 1)$ . Regularizing encoder with KL divergence, a S-VAE is trained with an autoencoder scheme. S-VAE has the ability to randomly generate sentence from standard distribution and to decode the change from one sentence to the other sentence.

While all techniques are similar to the origin variational autoencoder, the sequence-to-sequence version is suffered from the collapsing of latent space. The problem is similar to the concept of "cold start". At the start of training, decoder is random. Since the loss contains both KL divergence and language model, the KL divergence loss may first be optimized. That is, the encoder only outputs standard normal distribution. Language model cannot learn when the encoder is totally random. The proposed solution is simple: weighted the KL loss with the training step. At first, KL loss is muted, and gradually rises as the train process goes on. This is called KL annealing technique. Both linear and sigmoid function can aid S-VAE learning.

## 2.2 Prototype Editing

Based on the variation in sequence-to-sequence, neuron editor [7] set a mile stone in rephrasing sentences. First, it generates training pairs by selecting sentences with Jaccard distance  $< 0.5$ . Then, the model is trained with an edit vector. The edit vector contains the embedding of words to add and delete to the origin sentence, which is called prototype, and the vector predicts mean and standard deviation of normal distribution and then the vector is re-sampled from the distribution.

The way "prototype then edit" can not only rewrite sentences but also improve language model. With attention editing, neuron editor achieves lower perplexity. The prototype selection key in the whole process. First, it gives the problem into a supervised solution. Second, it lower the complexity of generating sentence. Last, it provides the edit directions.



## Chapter 3

# Problem Definition

In this section, we formally state the condition, explain the rephrasing goal and define the problem.

### 3.1 Definition

Our goal is to train a generative model that can rephrase sentence with given condition. Additionally, to avoid sophisticate labelling process, labelled corpus is not required. Instead, a condition function is used to tag each sentence. According to the previous description, we can formally state the problem. First, we introduce the notation.

- Corpus  $S = \{s_1, \dots, s_n\}$
- Condition Universe  $C_{All} = \{c_1, \dots, c_m\}$ , containing all possible condition of any  $s_i$
- Condition Function  $f : S \rightarrow C_{All}$
- Condition Set  $C = \{c_j | c_j = f(s_i), \forall s_i \in S\}$
- Generative Model  $g : (S, C) \rightarrow S$ , where  $S'$  is a generated sentence set and  $S' \neq S$

Then, a generative model  $g$  is required that generated sentence  $s' = g(s, c)$  should be similar to  $s$  and  $f(s') = c$ , regardless  $f(s) = c$  or not.

Additional Limit: The problem would be easy if there exists pairwise corpus. It would be a simple supervised task with  $(S, S')$  pairs. However, pairwise corpus seldom exists

for any condition function, so in this work, we focus on building models without pairwise corpus.



## 3.2 Example

To clarify the definition, we introduce an example in Table-3.1. First we select "best ice cream ever !" as the sentence to rephrase. Second, length function is used as the condition function and the origin condition is "length=5". Then, we select a target condition  $c = 10$ . Finally, we want the model to generate some sentences with given ("best ice cream ever !", 10), i.e. "the ice cream is the best i ever had !".

|                    |  |
|--------------------|--|
| Origin Sentence    | best ice cream ever !                  |
| Condition Function | length of the sentence                 |
| Origin Condition   | 5                                      |
| Target Condition   | 10                                     |
| Generated Sentence | the ice cream is the best i ever had ! |

Table 3.1: An example about length condition rephrasing



# Chapter 4

## Proposed Method

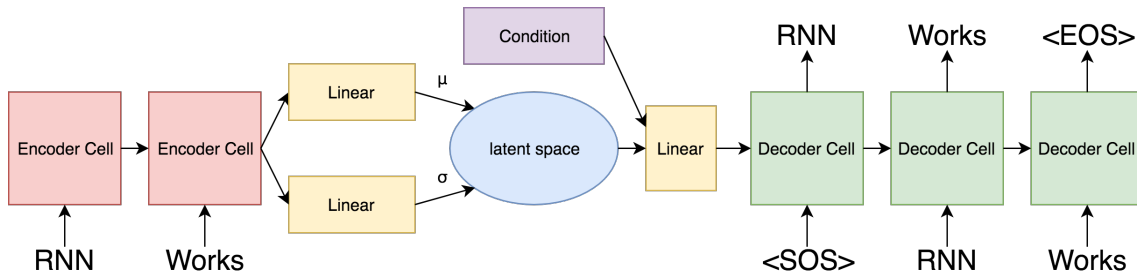


Figure 4.1: A Condition Variational Autoencoder Overview.

### 4.1 Conditional Sentence Variational Autoencoder

Based on Sentence-VAE[4], we propose a conditional variety, shown in figure 4.1. It is not a simple merging with Condition-VAE[14]. Instead, our model preserve encoder at testing phase and the type of condition is different from previous work. A CS-VAE has a encoder modeling posterior distribution  $q(\vec{z}|x)$  and a decoder modeling language model with given condition  $p(x|\vec{z}, c)$ .

At training phase, CS-VAE is trained as an autoencoder, as shown in figure 4.2, and the objective function is almost the same with Sentence-VAE using evidence lower bound (ELBO) to approximate the true likelihood of  $p(x)$ , which is

$$ELBO(\theta; x) = KL(q_{\theta}(\vec{z}|x)||p(\vec{z})) + E_{q_{\theta}(\vec{z}|x)}[p_{\theta}(x|\vec{z}, c)] \leq p(x) \quad (4.1)$$



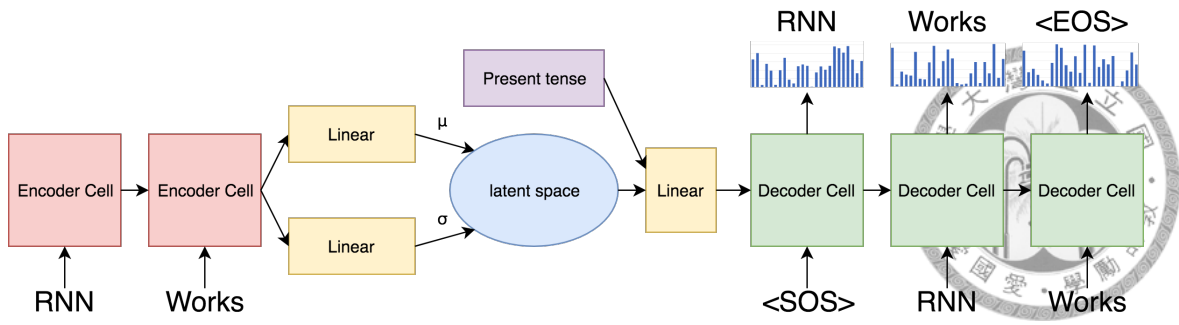


Figure 4.2: Training as an autoencoder.

Here,  $p(\vec{z})$  is normal distribution and since it is an autoencoder,  $c = f(x)$ . The ELBO requires KL annealing tricks mentioned in S-VAE[4] to optimized without collapsing in latent space. That is,  $q_{\theta}(\vec{z}|x) = 0$  to minimize KL loss, and the decoder always receives the same input from latent space. We adopt the sigmoid KL annealing process which the KL loss is weighted with a sigmoid function. At first, the weight is set to 0 and increases as the training step moves on. After an epoch, the KL weight rises to 0.5, which means that the function is actually goes like  $\sigma(\# \text{ of step} / \text{ steps per epoch})$ . After several epochs, the KL weight is close to 1. This trick lets decoder learn as much as it can, and then applies the loss to encoder. It prevents KL loss from collapsing because the likelihood would drop intensively if encoder collapse.

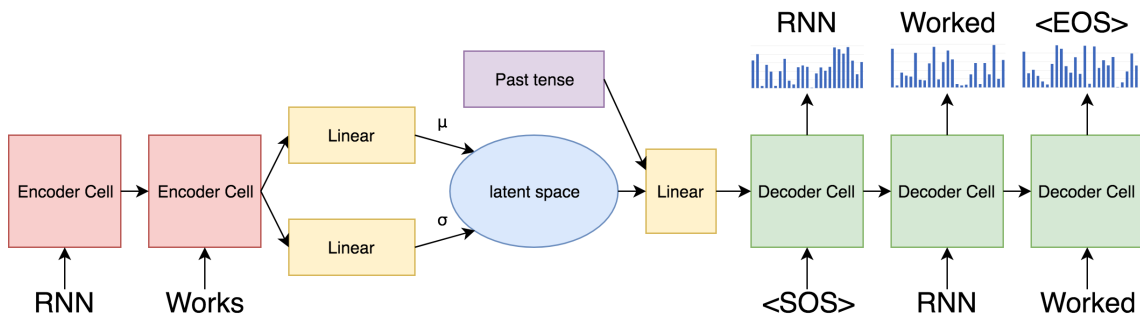
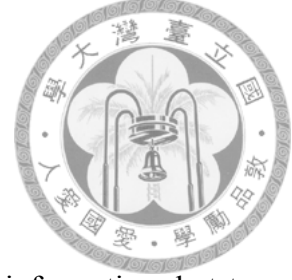


Figure 4.3: Training as an autoencoder.

At testing phase, CS-VAE works unlike other VAE approaches. It takes a sentence and a different condition as input, and then output a sentence similar to the origin sentence but matches the given condition. As the example in figure 4.3, if we train "RNN works" along with other present tense and past tense sentences, it may rephrase "RNN works" to "RNN worked" with the past tense condition. We have experimented the model on several conditions, and results strongly support that the rephrase sentences have high accuracy to

the given condition. Successfully achieving this goal, the condition mechanism is the most important trick, which will be discussed in the next sector.



## 4.2 Condition Mechanism

By design, the condition fed to decoder is not only to provide the information, but to eliminate the condition information in the latent space. Li et al. [11] shows that encoder has the ability to encode conditions to aid decoder such as length. If we explicitly provide the information, would the encoder still learn to encode such information? The answer varies to different models. For a simple sequence-to-sequence autoencoder, it still learns in encoder, but for a variational autoencoder, the answer is no. The variation of the latent space add some noise to the encoder information. The re-sample process adds uncertainty to decoder, so the decoder seeks for a more robust source, which is the fed condition. That is, the decoder first extract information from the condition, and then get the rest of the information from the encoder.

Looking back to the latent space, when some conditions are given explicitly, similar sentences with different condition may overlap in latent space. For example, given tense information, "RNN works" and "RNN worked" may fall into the same latent space because except one is present tense and the other is past tense, the meaning of the two sentences are the same.

We cannot actually force the similar sentence mapping into similar space, since the similarity relation remains unknown. But the unsupervised clustering can explain how we can manipulate different condition within the same sentence and the model can still output sentences with the correct condition and the similar semantic meaning.



## Chapter 5

# Experiments

In this chapter, we set up several experiments to exam the performance of our model in all dimension. First, the condition accuracy is experimented under several condition functions. Second, we exam the smoothness of the generated sentences. Third, the level of generative is evaluated. Finally, we justify the necessity of the variation part in our model.

### 5.1 Dataset

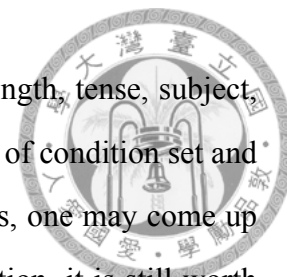
We report our result using Yelp open dataset [1], which contains millions of user’s review. The corpus is segmented into sentence level and then tokenized by NLTK package [3]. To avoid complex sentences, we select sentences with five to thirty tokens, which is a reasonable region to sentences without too many clauses. The corpus is cased to lowercase to prevent the emotional uppercase words from being new words in the vocabulary. After these preprocess steps, we split the corpus into training, validation and testing set. The size of each set is counted in table 5.1

|                      |           |
|----------------------|-----------|
| Training sentences   | 6,000,000 |
| Validation sentences | 500,000   |
| Testing sentences    | 744,962   |

Table 5.1: Number of sentences in each set

## 5.2 Condition Evaluation

In this section, we exam our model with four condition function: length, tense, subject, and single or plural. The four condition functions have different size of condition set and different distribution of the conditions. Although for some functions, one may come up with sophisticate rules to transform one sentence to the other condition, it is still worth evaluating the ability of our model, and the comparison to rule based method will not be discussed in this section. For each function, we will report the construction of the condition function, the distribution of each condition, the accuracy of every condition rephrasing to other conditions, and the samples of the generated sentences. The experiment will set based on the validation set.



### 5.2.1 Condition Function

In this section, we define the condition functions and show the distribution of each condition set.

- Length: the length of tokens of a sentence, and the condition set is length between 5 to 30
- Tense: First, sentence is labelled pos tag using NLTK [3]. Then, by capturing the existence of tag "VBP" and "VBZ", present tense is identified, and the "VBD" tag means the past tense. Future tense is captured by the word "will". Defining the condition, the condition set contains three conditions: "present tense", "past tense", and "future tense".
- Subject: First, sentence is parsed by spaCy2 [8]. Then, find the "nsubj" tag in the sentence. If the tag exists, sentences containing "I" or "We" are the first subject; sentences containing "You" are the second subject; other sentences are the third subject. But if there is no subject, it may be an imperative sentence or it is not a complete sentence. These sentences are classified into "other" category. To sum up, four conditions are in the condition set: "first subject", "second subject", "third subject", and "other".

- Single or Plural: First, a sentence is both processed by part-of-speech tagging and relative parsing. If a "VBZ" or "VBP" tag is found, one can identify whether the subject is single or plural. In other cases, we find the word with "nsubj" tag. If the word is "I", "It", "He", "She", "This", or "That", it has a single subject. And, if the word is "We", "They", "These", or "Those", it has a plural subject. Other sentences we cannot identify are tagged "other". Then, we have "single", "plural", and "other" in the condition set.

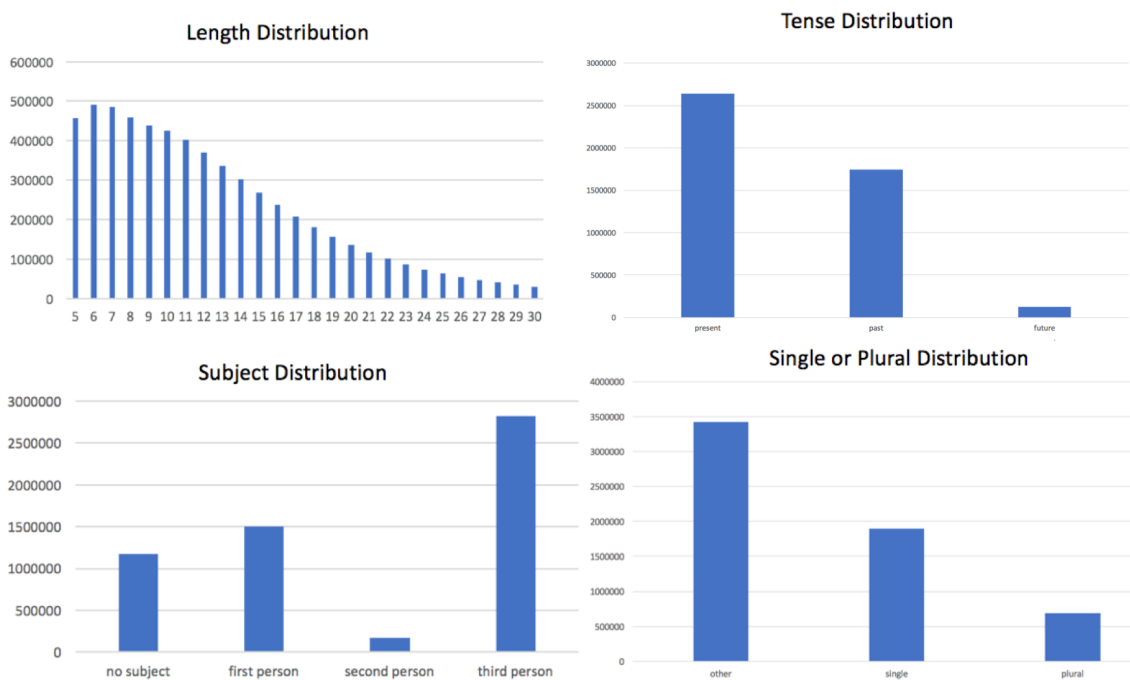


Figure 5.1: Length Distribution.

The figure 5.1 shows that the conditions are not uniformly distributed. Instead, for each condition, there are some condition with few instances. Since the conditions are different from labels, we do not apply data balance tricks and try to learn a model from the origin distribution.

## 5.2.2 Accuracy

Training the model to converge on the training set, we report the condition accuracy on the validation set. Since the length condition is continuous and the other three conditions are separated classes, we first discuss the length results and then the other three conditions.

First, the length condition accuracy is shown in figure 5.2. The table of numbers is placed in appendix since it is too complex. The figure shows that the accuracy is in-difference to the origin length. Even the autoencoder condition (the diagonal line) does not perform better than other conditions. Instead, the accuracy has higher relation to the length distribution. It is because for longer sentences, the training instance is fewer, and the model learns less to decode long sentences. Despite the long conditions have less accuracy, the overall accuracy is still high - the minimum is 0.84.

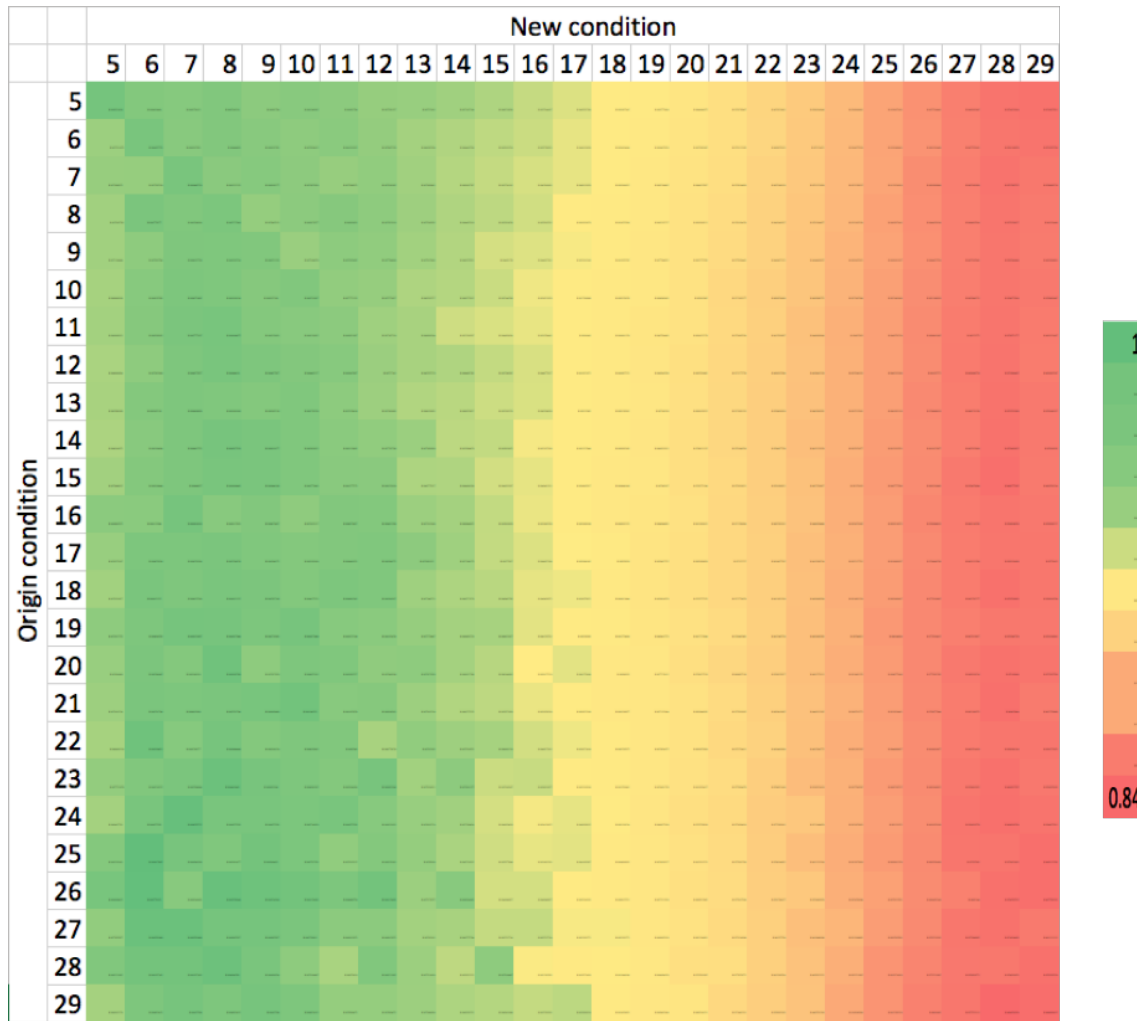
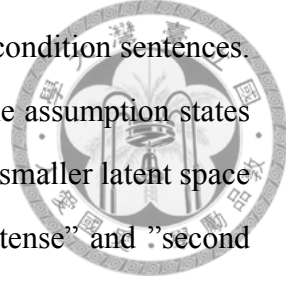


Figure 5.2: Length Accuracy.

The tense, subject, and "single or plural" condition results are shown in table 5.2, table 5.3, and table 5.4. Different from length condition, the same condition rephrasing gets better accuracy, and we observe that "future tense" and "second subject" get low accuracy when it is rephrased from other conditions. It is interesting that our model does

learn to decode the condition, otherwise the autoencoder score would not be high. But, the model cannot recognize the latent space encoded from the other condition sentences. We come up with several assumptions to this phenomenon. First, the assumption states that model learns few examples for the condition, so the model has a smaller latent space for the condition. Second, we observe that sentences with "future tense" and "second subject" are written differently to other conditions. The different distribution may result in the mismatch of latent space. The assumptions remain justified, and it is out of our topic, so we leave it as our future work.



| Origin Condition | New Condition |        |        |
|------------------|---------------|--------|--------|
|                  | Present       | Past   | Future |
| Present          | 0.9412        | 0.7980 | 0.5249 |
| Past             | 0.9112        | 0.9358 | 0.5667 |
| Future           | 0.8994        | 0.9238 | 0.9505 |

Table 5.2: Tense Condition Accuracy

| Origin Condition | New Condition |        |        |
|------------------|---------------|--------|--------|
|                  | First         | Second | Third  |
| First            | 0.9595        | 0.4439 | 0.8239 |
| Second           | 0.8802        | 0.8442 | 0.7838 |
| Third            | 0.8997        | 0.3688 | 0.8969 |

Table 5.3: Subject Condition Accuracy

| Origin Condition | New Condition |        |
|------------------|---------------|--------|
|                  | Single        | Plural |
| Single           | 0.9473        | 0.9197 |
| Plural           | 0.9088        | 0.9591 |

Table 5.4: Subject Condition Accuracy

### 5.2.3 Generated Sentences

In this section, we provide some generated sentences and their origin sentences to understand how the sentence is rephrased by our model in table 5.5, 5.6, 5.7, 5.8. Since the variational autoencoder always has a sampling part, the output is not deterministic. Good sentences and bad sentences can both be generated by our model, so we report a good and a bad generation for each input sentence.



|   |   |
|---|---|
| O | large salon with relaxing atmosphere (5)  |
| G | the atmosphere is relaxing and the place is comfortable (10) - Good                     |
| G | large family with a large group and my daughter . (10) - Bad                            |
| O | the staff here go out of their way to make sure your every need is taken care of . (19) |
| G | great attention to detail and take care of you . (10) - Good                            |
| G | the people here are to make sure their food . (10) - Bad                                |
| O | it was huge and filled with chopped bacon . (9)   |
| G | it was good and it came with the eggs and bacon wrapped in bread . (15) - Good          |
| G | it was so small and my mom were <unk> and the <unk> was filled . (15) - Bad             |
| O | the staff was attentive and very kind . (8)   |
| G | the staff was friendly and attentive and the food was good . (12) - Good                |
| G | the staff was attentive and very attentive and a very attentive . (12) - Bad            |

In the length examples, we observe that the generation can sometimes be precise in meanings while the other times it may be uncorrelated to the origin sentence or even the meaning is opposite.

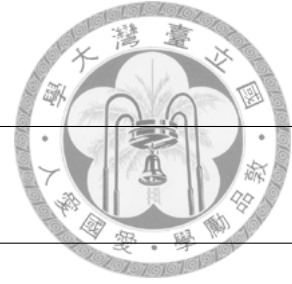
Table 5.5: Length Examples

|   |  |
|---|--|
| O | they are n't friendly but they get the job done . (Present)                    |
| G | they were n't friendly but they did the job ! . (Past) - Good                  |
| G | they were very friendly but the lady did n't work . (Past) - Bad               |
| O | my boyfriend & i were very happy to find this place . (Past)                   |
| G | my husband and i have found this place to be great . (Present) - Good          |
| G | my boyfriend & i go to this restaurant to <unk> food . (Present) - Bad         |
| O | i am back again .. (Present)   |
| G | i will be back again (Future) - Good   |
| O | food was good and the service was great ! (Past)                               |
| G | pizza was good and the service was great ! (Past Given Future Condition) - Bad |

For present and past tenses, the model gives pretty well results. Rephrasing to future tense, the third example "i am back again .." can be always transformed "i will be back again" or other similar sentences, so the bad generation cannot be provided. In contrast, the fourth example "food was good and the service was great" can never be rephrased into future tense, so no good generation is shown.

Table 5.6: Tense Examples





|   |   |
|---|---|
| O | i really like this place . (First)  |
| G | they really love this place . (Third) - Good  |
| G | it is like this place . (Third) - Bad   |
| O | food is good and the menu has a lot of good options . (Third)                             |
| G | i really enjoyed the food and the menu has a good variety . (First) - Good                |
| G | i really love the food and a lot of good options here . (First) - Good                    |
| O | if you 're in town you have to check it out (Second)                                      |
| G | i 'm in town we have to check it out . (First) - Bad                                      |
| O | we ate here with some friends a couple of weeks ago . (First)                             |
| G | we were here and a couple of friends on a saturday . (First Given Second Condition) - Bad |

The model generates well on the second example and is confused with the subject in the first example. The third and the fourth examples show the bad generation when the condition has relative to second subject.

Table 5.7: Subject Examples

|   |   |
|---|---|
| O | i have been going here for a few years . (Single)                               |
| G | we have been going here for a few years . (Plural) - Good                       |
| G | we ca n't wait to try a few drinks . (Plural) - Bad                             |
| O | they have a big menu selection and everything i order there was good . (Plural) |
| G | it was a big menu and the food was so good and fresh . (Single) - Good          |
| G | it was a big size and the staff was so friendly and helpful . (Single) - Bad    |
| O | i enjoyed the garlic wings the best . (Single)                                  |
| G | we had the best wings in town . (Plural) - Good                                 |
| G | we enjoyed the garlic knots the best . (Plural) - Bad                           |
| O | we had spicy ramen and traditional . (Plural)                                   |
| G | i had the ramen and spicy ramen . (Single) - Good                               |
| G | i had the spicy tuna roll . (Single) Bad  |

The generated sentences have correct condition, but sometimes the object is confused, such as garlic wings to garlic knots, spicy ramen to spicy tuna roll.

Table 5.8: Single or Plural Examples

## 5.3 Generation Quality

In this section, we report the quality of our model. To examine the quality of generated sentences, we first check the copy rate in table 5.9, which is defined as the percentage that the generated sentences are actually in the training set. Lower copy rate is better, which means that the generated space is not constrained by training data. Next, we evaluate the smoothness of the generated sentences by bi-gram model and tri-gram model in table 5.10. Despite the statistical language model cannot represent the overall quality of generation, we examine bi-gram and tri-gram probability for both training corpus and generated sentences. If the probabilities are close, it implies the generated sentences may have similar distribution to the training corpus. Last, we check if the generation is relative to origin sentence measured by Jaccard distance. The comparison baseline is defined as the expected jaccard distance if we randomly pick two sentences from the corpus. Repeating a million times, we randomly sample two sentences from the training set and then evaluate the Jaccard distance between the two sentences. As shown in table 5.11, the baseline is experimented to be 0.9361, and the distance between input sentence and the different condition generation is around 0.7 to 0.8. As expected, same condition rephrasing reaches lower distance. Overall, the distance results show that our generation is more similar to the input sentences.

| Condition Function | Same Condition | Different Condition |
|--------------------|----------------|---------------------|
| Length             | 0.0185         | 0.0066              |
| Tense              | 0.0888         | 0.0318              |
| Subject            | 0.0558         | 0.0370              |
| Single             | 0.0534         | 0.0407              |

Copy rate is very low for every circumstance.

Table 5.9: Copy Rate

## 5.4 Can An Autoencoder Also Work?

Claiming the proposed model works, we also check if a simple autoencoder with condition mechanism can work. Experimented with the length condition, condition autoencoder

| Condition Function | Bigram             | Trigram            |
|--------------------|--------------------|--------------------|
| Training sentences | $6.3639 * 10^{-8}$ | $4.6130 * 10^{-7}$ |
| Length             | $9.6287 * 10^{-9}$ | $7.1892 * 10^{-8}$ |
| Tense              | $6.4470 * 10^{-8}$ | $3.5980 * 10^{-7}$ |
| Subject            | $1.0933 * 10^{-7}$ | $5.8131 * 10^{-7}$ |
| Single             | $9.0479 * 10^{-8}$ | $5.0904 * 10^{-7}$ |



The generated sentences have similar probabilities to the training set.

Table 5.10: Bigram and Trigram Probability

| Random Sentences   | 0.9361         |                     |
|--------------------|----------------|---------------------|
| Condition Function | Same Condition | Different Condition |
| Length             | 0.7299         | 0.7869              |
| Tense              | 0.5993         | 0.7230              |
| Subject            | 0.6183         | 0.7429              |
| Single             | 0.6145         | 0.7396              |

Jaccard distance between generated sentences and the input sentences is significantly lower than random baseline.

Table 5.11: Jaccard Distance

fails to generate sentences with given length. As shown in figure 5.3, only the same length condition or the neighbor conditions have high accuracy. This shows that the decoder of an autoencoder cannot learn to use the explicit condition as the encoder information is directly passed.

| Old Condition | New Condition |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
|---------------|---------------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
|               | 5             | 6      | 7      | 8      | 9      | 10     | 11     | 12     | 13     | 14     | 15     | 16     | 17     | 18     | 19     | 20     | 21     | 22     | 23     | 24     | 25     | 26     | 27     | 28     | 29     |
| 5             | 0.8618        | 0.1356 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 6             | 0.0037        | 0.8735 | 0.1233 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 7             | 0.0000        | 0.0020 | 0.8818 | 0.1151 | 0.0001 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 8             | 0.0000        | 0.0000 | 0.0000 | 0.8996 | 0.1169 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 9             | 0.0000        | 0.0000 | 0.0000 | 0.0000 | 0.8785 | 0.1185 | 0.0001 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 10            | 0.0000        | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.8795 | 0.1188 | 0.0003 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 11            | 0.0000        | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.8846 | 0.1152 | 0.0006 | 0.0001 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 12            | 0.0000        | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.8847 | 0.1184 | 0.0011 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 13            | 0.0000        | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.8883 | 0.1256 | 0.0020 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 14            | 0.0000        | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.8926 | 0.1286 | 0.0036 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 15            | 0.0000        | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.8934 | 0.1454 | 0.0056 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 16            | 0.0000        | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.8956 | 0.1662 | 0.0109 | 0.0001 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 17            | 0.0000        | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.8958 | 0.2056 | 0.0173 | 0.0001 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 18            | 0.0000        | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.8971 | 0.2722 | 0.0294 | 0.0002 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 19            | 0.0000        | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.8999 | 0.3513 | 0.0405 | 0.0005 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 20            | 0.0000        | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.8958 | 0.4352 | 0.0550 | 0.0012 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 21            | 0.0000        | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.8965 | 0.5285 | 0.0724 | 0.0037 | 0.0002 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 22            | 0.0000        | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.8971 | 0.6211 | 0.1029 | 0.0097 | 0.0006 | 0.0000 | 0.0001 | 0.0000 |
| 23            | 0.0000        | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.8979 | 0.7099 | 0.1519 | 0.0144 | 0.0188 | 0.0009 | 0.0000 |
| 24            | 0.0000        | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.8979 | 0.7553 | 0.2225 | 0.0344 | 0.0039 | 0.0006 |
| 25            | 0.0000        | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.8980 | 0.8126 | 0.3282 | 0.0706 | 0.0124 |
| 26            | 0.0000        | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.8984 | 0.9035 | 0.4571 | 0.1422 |
| 27            | 0.0000        | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.8988 | 0.9298 | 0.6021 | 0.2872 |
| 28            | 0.0000        | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.8988 | 0.9494 | 0.8780 | 0.5088 |
| 29            | 0.0000        | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.8988 | 0.9548 | 0.9851 | 0.8988 |

Figure 5.3: Autoencoder Accuracy.



## Chapter 6

# Conclusions and Future Work

In this work, we first define a new generation problem, which cannot be solved by existing method. Next, a conditional sentence-VAE is proposed to rephrase an input sentence with a given condition. The CS-VAE works differently from other VAEs that it trains as an autoencoder, but at testing phase, with encoding the input sentence and a new condition, the model can generate a sentence satisfying the new condition while the meaning is similar to the input sentence.

Training with kl annealing tricks to converge, the model is examined with four condition functions. First, we define the four condition function: length, tense, subject and "single or plural". Then, the accuracy shows that our model can do well on condition generation, except the condition with too few examples. To solve the data unbalanced problem, the work is left as future work with possible solutions naming weighting techniques, feedback augmentation, and cross source learning. Last, we examine the copy rate and the bi-gram, tri-gram probability. Low copy rate and similar probability score support the generate sentences have good quality.

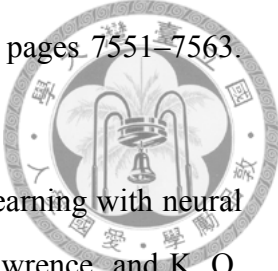


# Bibliography

- [1] Yelp open dataset. <https://www.yelp.com/dataset/>.
- [2] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015.
- [3] S. Bird, E. Klein, and E. Loper. Natural Language Processing with Python. O’Reilly Media, Inc., 1st edition, 2009.
- [4] S. R. Bowman, L. Vilnis, O. Vinyals, A. Dai, R. Jozefowicz, and S. Bengio. Generating sentences from a continuous space. In Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning, pages 10–21, Berlin, Germany, Aug. 2016. Association for Computational Linguistics.
- [5] C. K. Chen, Z. F. Pan, M. Sun, and M. Liu. Unsupervised stylish image description generation via domain layer norm. CoRR, abs/1809.06214, 2018.
- [6] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2, NIPS’14, pages 2672–2680, Cambridge, MA, USA, 2014. MIT Press.
- [7] K. Guu, T. B. Hashimoto, Y. Oren, and P. Liang. Generating sentences by editing prototypes. CoRR, abs/1709.08878, 2017.

- [8] M. Honnibal and I. Montani. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear, 2017.
- [9] A. Karpathy, J. Johnson, and F. Li. Visualizing and understanding recurrent networks. CoRR, abs/1506.02078, 2015.
- [10] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings, 2014.
- [11] J. Li, X. Chen, E. Hovy, and D. Jurafsky. Visualizing and understanding neural models in NLP. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 681–691, San Diego, California, June 2016. Association for Computational Linguistics.
- [12] T. Luong, H. Pham, and C. D. Manning. Effective approaches to attention-based neural machine translation. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1412–1421, Lisbon, Portugal, Sept. 2015. Association for Computational Linguistics.
- [13] D. Rezende and S. Mohamed. Variational inference with normalizing flows. In F. Bach and D. Blei, editors, Proceedings of the 32nd International Conference on Machine Learning, volume 37 of Proceedings of Machine Learning Research, pages 1530–1538, Lille, France, 07–09 Jul 2015. PMLR.
- [14] K. Sohn, H. Lee, and X. Yan. Learning structured output representation using deep conditional generative models. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, Advances in Neural Information Processing Systems 28, pages 3483–3491. Curran Associates, Inc., 2015.
- [15] S. Subramanian, S. R. Mudumba, A. Sordoni, A. Trischler, A. C. Courville, and C. Pal. Towards text generation with adversarially learned neural outlines. In S. Ben-

gio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 7551–7563. Curran Associates, Inc., 2018.

- 
- [16] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc., 2014.
- [17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.
- [18] L. Yu, W. Zhang, J. Wang, and Y. Yu. Seqgan: Sequence generative adversarial nets with policy gradient. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, AAAI’17*, pages 2852–2858. AAAI Press, 2017.