國立政治大學應用數學系
碩士學位論文

跨語言遷移學習在惡意留言偵測上的應用
Cross-lingual Transfer Learning for Toxic Comment Detection

國立政治大學 National Chengchi University

碩士班學生：陳冠宇　撰
指導教授：蔡炎龍　博士
中 華 民 國　109　年　8　月

DOI:10.6814/NCCU202001728

致謝

來政大原來已經兩年了，卻覺得從大學畢業是昨日的事而已，很開心自己能來到政大應數這個大家庭。

謝謝炎龍老師願意收我當學生當我的指導教授，在完成這篇論文的過程中幫助我給了我許多建議，讓我能順利的完成這篇論文; 謝謝老大在學校的照顧，不僅在數學上加深了我的底子，也加深了我應酬的能力; 謝謝張媽，符媽讓我擔任助教，並信任我把每一件事都交給我處理; 謝謝澤佑學長，瑄正在我碩二時給了我許多人生與論文上的建議; 謝謝瑞鋆在課業上或生活上都帶給我許多幫助，我等著你去美國太空總署我會去找你的; 謝謝冠棋與承孝帶我加入了球隊，也豐富了我碩士的生活帶來了許多歡笑; 謝謝凱瑩在我碩士時陪伴著我，跟我一起努力; 謝謝研究生室的每一位，這個大家庭有著許多有趣的回憶。最後感謝我的家人，感謝你們在背後默默的支撐著我，讓我有著動力一直前進，並且最後完成對自己的期許，謝謝我在碩士中遇到每一位的人事物讓我有了成長。

國立政治大學 National Chengchi University

DOI:10.6814/NCCU202001728

中文摘要

Transformer 這個模型，它開啟了自然語言處理領域的一道大門，使得這個領域往前邁進了一大步，它讓模型更了解了文字中的關係。並且它的模型架構延伸了許多語言模型，例如跨語言模型的 XLM，XLM-R，而這些延伸出來的模型在各個任務中都獲得了很好的成績。在本篇論文中，我們證實了可以透過其他高資源的語言來彌補低資源的語言的資料量，我們以預測留言是否是惡意留言來做為例子，我們分別使用 Jigsaw Multilingual Toxic Comment Classification 競賽所釋出的英文資料和 PTT 黑特版上的留言當做輸入的訓練集，並要模型預測中文的惡意留言，而且英文的資料量比中文的資料量多出很多，我們將其預測結果分為三個種類分別是單純以英文資料訓練模型，單純以中文資料訓練模型，最後是將兩者的資料結合並訓練模型，發現在以英文資料的訓練因為其資料量較大使得其預測結果為最好有 75.9% 的水準，而以總體預測水準來說為混合型的資料分數較高有 88.3%。總體來說，我們可以透過跨語言模型來補足低資源語言的不足，並且有了另一種解決低語言資料的方法。

關鍵字: Transformer，XLM-R，跨語言預測，惡意留言，不平衡數據，深度學習，對話安全

國立政治大學 National Chengchi University

DOI:10.6814/NCCU202001728

Abstract

The Transformer model, which opens a door in the field of natural language processing, makes this field has another significant further step. It allows the model to better understand the relationship in the word. And the model architecture extends many language models, such as cross-lingual model XLM, XLM-R, and these models have achieved good results in various tasks. In this paper, we proved that other high-resource languages can be used to make up for the data in low-resource languages. We take the prediction of whether the comment is a toxic message as an example. We use the English data released by the Jigsaw Multilingual Toxic Comment Classification competition and the comment on the PTT Hate board as the input training set. We want the model to predict toxic comment in Chinese, and the data in English is much larger than that in Chinese. We divide the prediction results into three categories: only use English data to fine-tune the model, fine-tune the model with Chinese data, and the last is combine the two data and fine-tune the model. We found that the training with English data has the best accuracy score of 75.9% because of the large amount of data, while the overall accuracy scores that mixed data has a higher score of 88.3%. In general, we can make up for the lack of low-resource languages through cross-lingual models and have another way to solve low-resource languages problem.

Keywords： Transformer, XLM-R, cross-lingual prediction, toxic comment, imbalanced data, deep learning, security conversactions

國立政治大學 National Chengchi University

DOI:10.6814/NCCU202001728

Contents

致謝 **i**

中文摘要 **ii**

Abstract **iii**

Contents **iv**

List of Figures **vi**

1 Introduction **1**

2 Deep Learning **3**

2.1 Neurons and Neural Networks 4

2.2 Activation Function 6

2.3 Loss Function 8

2.4 Gradient Descent Method 9

3 Transformer **11**

3.1 Structure of Transformer 11

3.2 BERT 15

3.3 RoBERTa 18

3.4 Unsupervised Cross-lingual Representation Learning at Scale: XLM-R 19

4 Cross-language prediction and Imbalanced Data **22**

4.1 Toxic Message 22

4.2 Imbalanced Data 23

國立政治大學 National Chengchi University

DOI:10.6814/NCCU202001728

4.2.1 Data-level methods 23

5 Experiments **25**

5.1 Prediction model 26

5.2 English training dataset 26

5.2.1 Random-oversampling(ROS) 27

5.2.2 Random-undersampling(RUS) 27

5.3 Chinese training dataset 28

5.3.1 Random-oversampling(ROS) 28

5.3.2 Random-undersampling(RUS) 29

5.4 Combined training dataset 29

5.4.1 Random-oversampling(ROS) 29

5.4.2 Random-undersampling(RUS) 30

5.4.3 Adjust ratio of sample 30

6 Conclusion **31**

Bibliography **33**

國 立 政 治 大 學
National Chengchi University

DOI:10.6814/NCCU202001728

List of Figures

2.1 Three steps of deep learning 4
2.2 The Structure of a Neuron 5
2.3 Fully Connected Feedforward Network 6
2.4 Rectified linear unit (ReLU) 7
2.5 Sigmoid function 7
2.6 Hyperbolic tangent (tanh) 8
2.7 Gradient Descent 10

3.1 Structure of Transformer 13
3.2 pre-training of BERT 16
3.3 5 steps of fine-tuning process 17
3.4 Single Sentence Classification Tasks 18
3.5 Comparison Table of BERT and RoBERTa 19
3.6 Translatoin Language Modeling(TLM) 20

4.1 Random oversampling(ROS) 24
4.2 Random undersampling(RUS) 24

5.1 training dataset of English 25
5.2 training dataset of Chinese 25
5.3 testing dataset of Chinese 26
5.4 Average accuracy of M_o 26
5.5 Average accuracy of M_o and M_{ro} 27
5.6 Accuracy of M_{ru} 27
5.7 Average accuracy of M_c 28

國立政治大學 National Chengchi University

DOI:10.6814/NCCU202001728

5.8 Average accuracy of M_{cro} 28
5.9 Average accuracy of M_{cru} 29
5.10 Average accuracy of M_{mix} 29
5.11 Average accuracy of M_{mru} 30
5.12 Average accuracy of M_{mru} 30
5.13 Average accuracy of M_{msp} 30

6.1 Average accuracy of all model 31

國立政治大學 National Chengchi University

DOI:10.6814/NCCU202001728

Chapter 1

Introduction

In these years, artificial intelligence has become one of the hottest fields. Since technology persistently and constantly making progress that make computing power of computers has greatly improved, which also make us to greatly shorten the time to train the model. Machine learning is a method of artificial intelligence, and one of the most commonly used in machine learning is deep learning [13]. Deep learning have lots of applications, and it is divided into three major applications: image recognition [8], speech recognition [21], natural language processing [2]. In this paper, the topic we are discussed that belongs to the field of natural language processing. In 2017, Google publish the state-of-the-art of language models that name is called "Transformer" [18]. Then in 2019, Facebook present "XLM-R" which have the state-of-the-art performance on cross-lingual benchmarks [4]. We want to use XLM-R to predict cross-lingual task. Specifically, we want to use english training set to predict chinese testing set.

Social media give us an environment where we can freely discuss and share their thought to others. But it is a pity that have some people who bring a negative impact on the environment [9]. They maybe use fake accounts and leave messages to harass others, or use words to bully or abuse others. Because of these problems, the user experience of social media will get worse. Hence, we want to improve the safety of the online environments, therefore we use deep learning to detect toxic message. In real life, there are more general comment than toxic comment, which leads to the dataset is imbalanced. We use two simple method to solve this problem that is ROS [7] and RUS [6]. Since imbalance data will cause the accuracy of smallest(minority) class have under performance [20], the unbalanced ratio of our english training set is $\rho = 10$ and the

DOI:10.6814/NCCU202001728

chinese training set is $\rho = 5.16$, and the largest class is 0 that mean the comment is not toxic message, and the smallest class is 1 which mean the comment is toxic message.

If the target language resources you want to predict are low-resource, we will present that you can use high-resource language to supplement the set of low-resource language to enhance the accuracy of the task. In this paper, we will compare the prediction result from English, Chinese and combine the both, which to observe the performance.Our code is put on https://github.com/Kyle010166/thesis

國立政治大學 · National Chengchi University ·

DOI:10.6814/NCCU202001728

Chapter 2

Deep Learning

Let computer like human neural and learning by itself this called deep learning. It’s easily to know deep learning, it a best function find in the function set, we give it input data and output a lot of values, and best function calculated by the machine. Below was told you how the function work.

$$f(\text{“ I am very happy ”}) = \text{“ positive sentence ”}$$

When we use deep learning in classification, we want machine know what the sentence mean. If we set the sentence “I am very happy.” Then function will give you classification is “positive sentence.” Above example you input your current sentence and computer will identify the sentence then give you classification.

We will divide deep learning into three stepss construct the model, find the right function and then predict your goal. The concept of deep learning is to train the model through a large amount of data to tell the model the relationship between input and output. And the result of this model is whether or not it is good, we need to design a judgment standard. According to the judging criteria then choose the best model.

DOI:10.6814/NCCU202001728

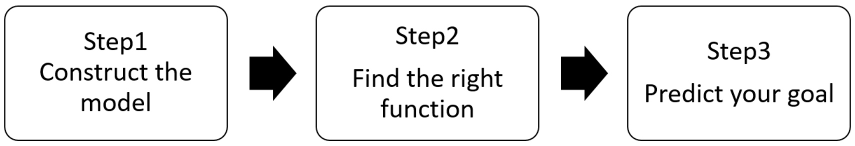


Figure 2.1: Three steps of deep learning

2.1 Neurons and Neural Networks

Artificial neural networks and human brains really have some similarities, we all know that the human brain is made up of neurons, and the network is also connected by “neurons.” [10] This section will introduce the structure of neural networks and how to imitate human neurons.

Every basic neurons is a function and its symbolσwe called it activation function in deep-learning, the activation function is a non-linear function defined by human in advance. Each input has a corresponding weight and multiplying each to get the value therefore the input to these functions is a set of values (e.g. is a vector) then the output is a value. As in Figure 2.2, the left side are $x_1,\ x_2,\ x_3$ are the input of neuron and $w_1,\ w_2,\ w_3$are the weight. And right side, h is the output of neuron. The bias of the neuron at below of the Figure 2.2 is b, and the function of neurons σ is $h = \sigma(\sum_{i=1}^{n} w_i x_i + b)$.

DOI:10.6814/NCCU202001728

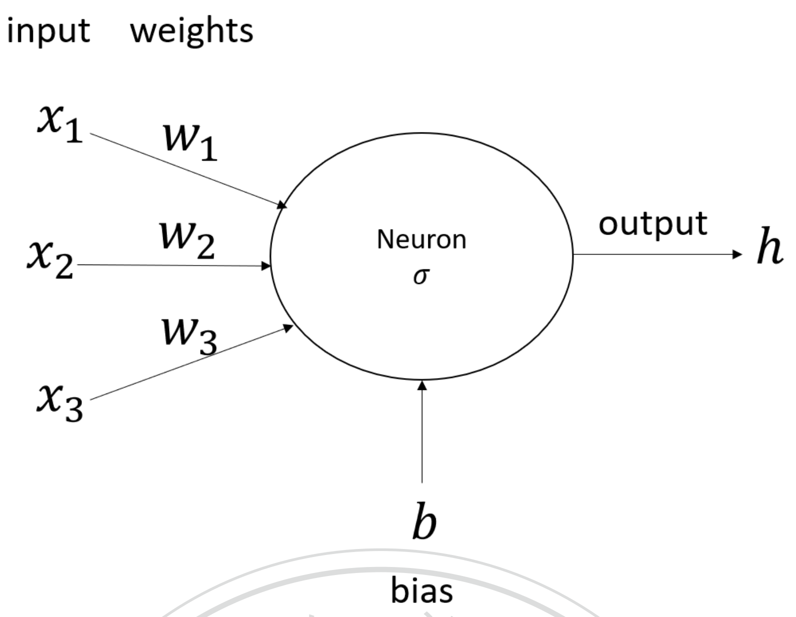


Figure 2.2: The Structure of a Neuron

After understanding neurons, we started to introduce the process of neuron works. Beginning we let input values multiplied by the corresponding weights. Then sum up product values and bias value. Next, the value through activation function we get output value, this is the complete process of the neuron. Giving an example to clearly understand the process. We set $x_1,\ x_2,\ x_3,\ w_1,\ w_2,\ w_3,\ b$ according to the order is $2,\ (-1),\ 2,\ 1,\ (-1),\ 2,\ 1$ and the equation is $2 \times (1) + (-1) \times (-1) + 2 \times 2 + 1 = 7$ and the input to the activation function is 7. Setting the activation function is ReLU which will introduce the details in later chapter. Hence ReLU received value 7 and output is 7. Because when the input is less than 0,the output is 0; when the input is greater than 0, the output equals input as below function

$$\sigma(2 \times (1) + (-1) \times (-1) + 2 \times 2 + 1) = \sigma(-7) = 7.$$

The wights and bias in neurons are called parameters. They determine how neurons work, when the machine is training, these parameters are automatically adjusted according to the training data.

Let we look at neuron network that made up of many neurons, and we only need to decide how to connect neural networks . We give an example as figure2.3 that is a kind of model of neuron network in deep learning.

DOI:10.6814/NCCU202001728

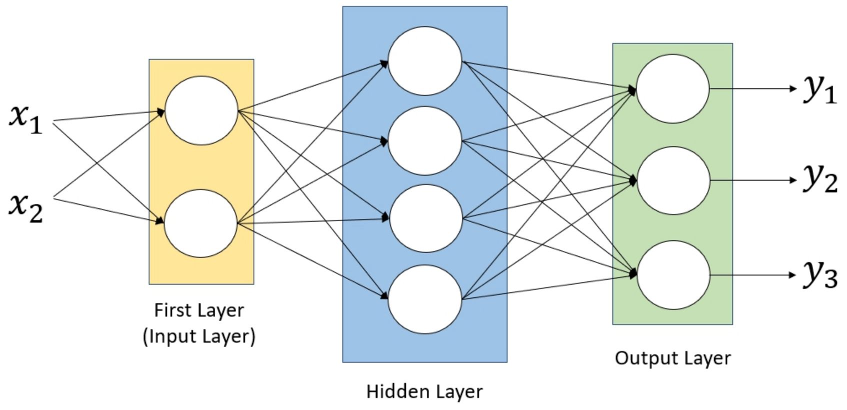


Figure 2.3: Fully Connected Feedforward Network

In figure2.3 we called it is a fully connected feedforward network that was the first and simplest type of artificial neural network [16]. Its working method is very easy. The neurons of the first layer have obtained the features value, multiplied by the W weight plus bias deviation, and the result is calculated through the activation function and passed to the next layer, so Feedforward Neural Networks is a unidirectional flow. Does not return the value from the output to the input. The first layer we also called input layer and the last layer called output layer. Between input and output layer are known as hidden layers, as the training data does not show the desired output for these layers. A network can contain any number of hidden layers with any number of hidden units.

2.2 Activation Function

Use activation function in neuron networks, we mainly using nonlinear equations to solve nonlinear problems. Since the hidden layer and the output layer input the result of the upper layer and are linear combination of the input. In reality, many problems are nonlinear problems. At the below, we show three most used activation function in detail.

1. Rectified linear unit (ReLU)

   Equation:

$$f(x) = \begin{cases} x, & \text{if} \quad x \geq 0 \\ 0, & \text{if} \quad x < 0 \end{cases}$$

   Range: $[0, \infty)$

DOI:10.6814/NCCU202001728

Graph:

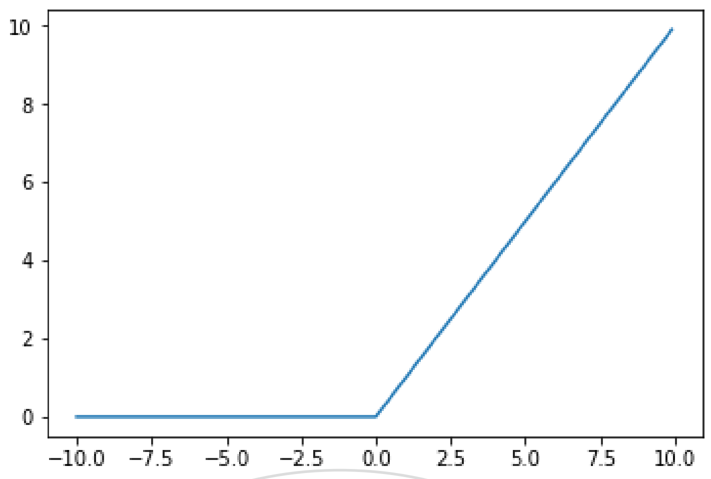


Figure 2.4: Rectified linear unit (ReLU)

2. Sigmoid function

   Equation:

$$f(x) = \frac{1}{1+e^{-x}}$$

   Range: $(0, 1)$

   Graph:

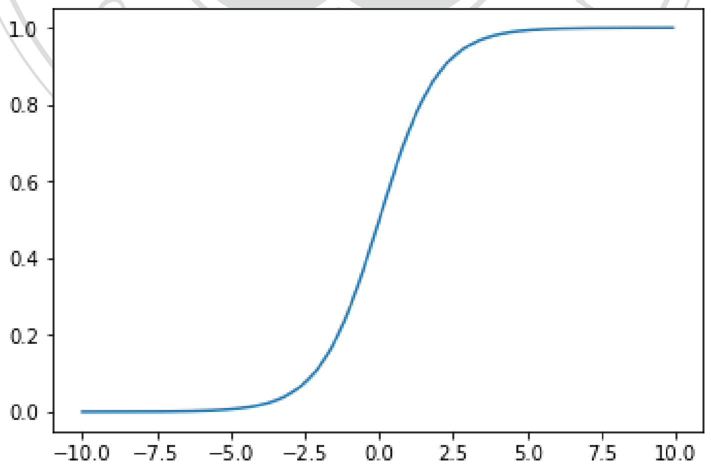


Figure 2.5: Sigmoid function

3. Hyperbolic tangent (tanh)

   Equation:

DOI:10.6814/NCCU202001728

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

Range: $(-1, 1)$

Graph:

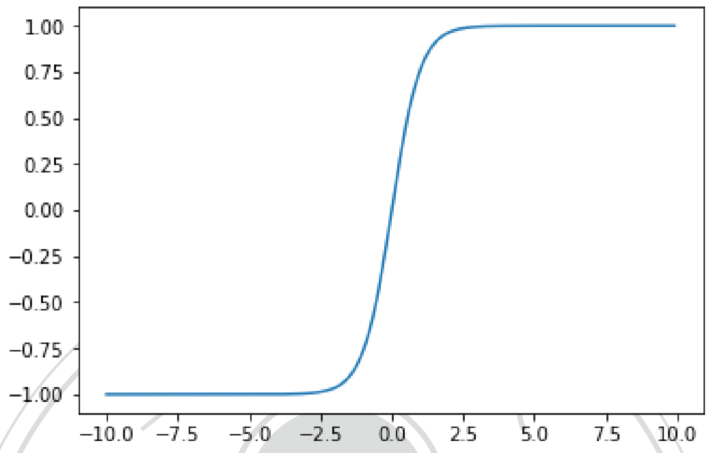


Figure 2.6: Hyperbolic tangent (tanh)

2.3 Loss Function

We mentioned earlier that looking for the best parameter among the parameter space, but to compare the quality of each parameter, we need a judgment criterion. We called this criterion is loss function. The loss function estimates how much the prediction deviates from the actual value and this difference can called it error. But in each training, we will get a bunch of errors. Hence, we need to choose a correct loss function. The following are three most simple loss functions.

We define $\mathbf{y}_i$ represents the actual value, $\hat{\mathbf{y}}_i$ represents the predicted value and k is numbers of data .

1. Mean absolute error (MAE)

$$MAE = \frac{1}{k}\sum_{i=1}^{k} ||\mathbf{y}_i - \hat{\mathbf{y}}_i||$$

DOI:10.6814/NCCU202001728

2. Mean squared error (MSE)

$$MSE = \frac{1}{k}\sum_{i=1}^{k} ||\mathbf{y}_i - \hat{\mathbf{y}}_i||^2$$

3. Binary cross-entropy(H)

$$H = -\frac{1}{k}\sum_{i=1}^{k}[\mathbf{y}_i \log(\hat{\mathbf{y}}_i) + (1 - \mathbf{y}_i)\log(\hat{\mathbf{y}}_i)]$$

2.4 Gradient Descent Method

In deep learning, no matter what kind of network. We need to find the relationship of parameters between layers, and the process of finding parameters is called learning. So the purpose of the neural network is to constantly update the parameters then minimize the value of the loss function and find the best solution

The "gradient descent" is one of the methods in the optimization theory to constantly update the parameters to find a solution. We according the parameters of above neural networks define $\theta = \{w_1, w_2, ..., w_n, b_1, b_2, ..., b_m\}$, we hope we can find the optimal solution θ^* such that minimizes the loss function $L(\theta)$. In this method, machine assign a random value as an initial value for w_1 denoted as $w_1^{(1)}$ then compute its first derivation $\frac{\partial L}{\partial w_1^{(1)}}$ then update to the better parameters i.e.

$$w_1^{(2)} = w_1^{(1)} - \eta\frac{\partial L}{\partial w_1^{(1)}}$$

which η is learning rate and will introduce later. Use this way until $\frac{\partial L}{\partial w_1}$ is approach to zero. Others parameters are the same process to find the best solution, therefore the iteration is like the below in figure 2.7 and we will repeat this process until all the parameter of norm is small enough.

DOI:10.6814/NCCU202001728

$$\begin{bmatrix} w_1^{i+1} \\ w_2^{i+1} \\ \vdots \\ w_n^{i+1} \\ b_1^{i+1} \\ b_2^{i+1} \\ \vdots \\ b_m^{i+1} \end{bmatrix} = \begin{bmatrix} w_1^{i} \\ w_2^{i} \\ \vdots \\ w_n^{i} \\ b_1^{i} \\ b_2^{i} \\ \vdots \\ b_m^{i} \end{bmatrix} - \eta \begin{bmatrix} \frac{\partial L}{\partial w_1^i} \\ \frac{\partial L}{\partial w_2^i} \\ \vdots \\ \frac{\partial L}{\partial b_m^i} \end{bmatrix}$$

Figure 2.7: Gradient Descent

The learning rate is a hyperparameter that grasp the learning progress of the model that affects how fast our model can coverge to a local minimum. Generally, the greater the learning rate, the faster the neural network learns. If the learning rate is too small, the network is likely to fall into a local optimum; but if it is too large and exceeds the extreme value, the loss will stop decreasing and oscillate repeatedly at a certain position.

國立政治大學 National Chengchi University

DOI:10.6814/NCCU202001728

Chapter 3

Transformer

Nowadays, in the field of Natural Language Processing(NLP), nobody does not know about Transformer(Vaswani et al., 2017) [18]. Transformer contribution is the self-attention mechanism in the structure of sequence to sequence model (Seq2Seq) [18]. Sequence to sequence model is composed of encoder and decoder which are sequential model, and the sequential maodel is good at processing data with sequence characteristics. Therefore, we input a sequence and then encoder will convert the sequence into a vector, usually we call it a context vector. then decoder generates text based on context vector. And self-attention mechanism improves the previous problems of the Seq2Seq model.

國立政治大學 National Chengchi University

3.1 Structure of Transformer

First, let talk about what is embedding. Computer is made up of numbers, so that is more easier receive the numbers then word. Therefore, we need to covert the word to numerical data. Suppose have a sentence is "apple is a kind of fruit" then we put it into dictionary that becomes ["apple", "is", "a", "kind", "of", "food"], and we can make each word correspond to a vector so "apple" is correspond to $[1, 0, 0, 0, 0, 0]$ and "of" is correspond to $[0, 0, 0, 0, 1, 0]$, this method is called one-hot ecoding.

Transformer use the method is called wordpiece embedding that is divides every words to sub-word units and switch them to index sequence. [19] For example, we have a word "apple" we divide it to "app"and "le" then the sequence will be $[1, 2]$. And this method has a benefit that can easily compose the word. There have four index in the Transformer one is the "start

DOI:10.6814/NCCU202001728

of sequence” (<SOS>) which is the represent beginning of the sentence, second is “end of sequence” (<EOS>) which is represent the end of the sentence, third is “padding mask” which function is because the length of each batch of input sequence will be different, and we need to use it to make each input sequence be the same length. Specifically, it is to fill 0 up after the shortest sequence. Since these positions are actually meaningless, our attention mechanism should not focus on these positions and we will introduce attention mechanism in this section later. And the other index we will introduce at the end.

In addition to word embedding, because the Transformer cannot extract sequence order information, so Transformer need to add the information of position. To solve this problem, Transformer uses positional embedding that encode the position of the word in the sequence and the value will be

$$PE(pos, 2i) = \sin(\frac{pos}{10000^{\frac{2i}{d_{model}}}})$$

$$PE(pos, 2i-1) = \cos(\frac{pos}{10000^{\frac{2i-1}{d_{model}}}})$$

where d_{model} is the model dimension in the original paper it set $d_{model} = 512$,$1 \leq i \leq \frac{d_{model}}{2}$ and $1 \leq pos \leq d_{model}$. Then word embedding sum with positional embedding that can extract the word information and order information.

Now, We introduce that the Seq2Seq model is input a sequence and output is also a sequence. Seq2Seq model can divided into architectures which are encoder and decoder that can contain blocks with the same multi-layer structure. In Transformer each layer will have multi-head attention and feed forward network. The encoder will converts the input sequence $(x_1, x_2, ..., x_n)$ to $(z_1, z_2, ..., z_n)$ and the decoder converts $(z_1, z_2, ..., z_n)$ to $(y_1, y_2, ..., y_n)$ that convert one at a time, but when decoder attend encoder output need mask to avoid to see future information. For example, like language translation we input the processed sentence in encoder then we can get hidden vectors which will input in decoder and translate to target language. In previous Seq2Seq models, Recurrent Neural Network(RNN) used to construct Seq2Seq model, since RNN which is often used to solve the sequence problem. But there is a disadvantage that cannot effectively parallel operation. It means we get the output vector y_n at the last time point, it had to run the entire input sequence. Self-attention mechanism solve cannot effectively parallel operation problem also can solve the sequence problem. The architecture of Transformer as Figure 3.1.

DOI:10.6814/NCCU202001728

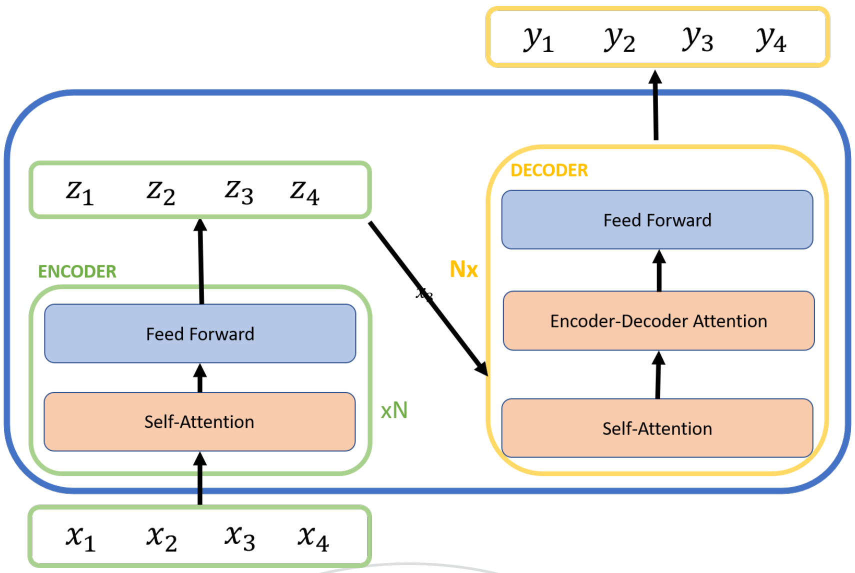


Figure 3.1: Structure of Transformer

The help brought by the self-attention mechanism we mentioned earlier. Now, we will show it how to works. When Self-attention performing sequence generation tasks such as machine translation, we need to add additional Positional Encoding to add sequence information. Self-attention layer will take each word vector x_i or have some articles called representation which through three linear transformations to three vectors q_i ﹐ k_i and v_i get a example at below. Let x_1, x_2, x_3 to x_n be n input word vectors and all dimension will not exceed d_{model}. And the equation is define as:

$$Q: \text{ query (to match others)}$$

$$q_i = W^Q \cdot x_i$$

$$K: \text{ key (to be matched)}$$

$$k_i = W^K \cdot x_i$$

$$V: \text{ information to be extracted}$$

$$v_i = W^V \cdot x_i$$

where $1 \leq i \leq n$, W^K and $W^Q \in \mathbb{R}^{d_k \times d_{model}}$, $W^V \in \mathbb{R}^{d_v \times d_{model}}$. Since W^K, W^Q and W^K is the metrics that Q have same dimension as K, then d_k is dimension of key and d_v is dimension of value.

Each q vectors matches k at other position in the sequence then after calculating the self-

DOI:10.6814/NCCU202001728

attention weights, use the softmax layer to obtain a weight value between 0 and 1. Continuing we use value to weights average with v. Hence, we get the output vector b_i at the position i. All of output vector can effectively parallel operation and obtain the output sequence. The original author uses following function Scaled Dot-Product Attention to get the weight. The function will show as below:

$$a_{1,i} = \frac{q^1 \cdot k^i}{\sqrt{d}}, \text{where d is the dimension same as q and k,and } 1 \leq i \leq n$$

then we get

$$\hat{a}_{1,i} = \frac{\exp(a_{1,i})}{\sum_{j=1}^{n} \exp(a_{1,j})}$$

$$b_1 = \sum_{i=1}^{n} \hat{a}_{1,i} \cdot v_i$$

Now, we have 1_{st} position output b_1 which is a vector and we can get b_2, b_3, to b_n at the same way. Then we use x_i to consist a matrix A which mean x_1 vector is the first column of matrix A and x_2 vector is the second column of matrix A and so on, q_i, k_i and v_i all use the same method to consist matrixs Q, K and V. Hence, the function of matrixs can be shown as bellow:

$$Q = W^Q A$$

$$K = W^K A$$

$$V = W^V A$$

$$\text{Attention}(Q, K, V) = [\text{softmax}(\frac{Q^T K}{\sqrt{d_k}})]^T V^T$$

where W^K and $W^Q \in \mathbb{R}^{d_k \times d_{model}}$, $W^V \in \mathbb{R}^{d_v \times d_{model}}$, d_k is dimension of key and d_v is dimension of value and $P \in \mathbb{R}^{I \times I}$ we denote I is the maximun length of sequence and P is the matrix of padding mask. The attention mask in the encoder of Transformer is always be padding mask.

Now, we will cut the Q, K, V of each position in the input sequence into multiple q_{ij}, k_{ij}, v_{ij} for i is the i_{th} position and j is the number of head, and then separately to do self-attention which we called this method is Multi-head Attention. After each processing, concatenate all the results and reduce the dimension according to the situation. The advantage of this is that each head can perform its duties and learn to attend to the information in different representation

DOI:10.6814/NCCU202001728

spaces in different positions in the sequence. We show the function as follow:

$$\text{Multihead}(Q, K, V) = W^O[\text{Concatenate}(\text{head}_1, \text{head}_2, ..., \text{head}_h)]^T$$

$$\text{where head}_j = \text{Attention}(Q \cdot W_j^Q, K \cdot W_j^K, V \cdot W_j^V)$$

where W_j^K and $W_j^Q \in \mathbb{R}^{d_{model} \times d_k}$, $W_j^V \in \mathbb{R}^{d_{model} \times d_v}$, $W^O \in \mathbb{R}^{d_{model} \times hd_v}$ for $1 \leq j \leq h$, and Concatenate is the function that introduce as above. Google sets $h = 8$ parallel heads, and for each we use $d_k = d_v = 64$.

In the decoder of transformer have the index tokken it called "sequence mask" which is to make the decoder unable to see future information. For a sequence at the time-step t and our decoder output should only depend on the output that before time t, and not on the output after t. Therefore, we need to hide the information after time-step t. Then the attention mask in the decoder of transformer we need to sum with padding mask and sequence mask.

3.2 BERT

Bidirectional Encoder Representations from Transformers we call it BERT for short. BERT is a language representation model generated by (Devlin et al., 2018) [5] use unsupervised way and a large amount of text without annotation. BERT structure is encoder of Transformer.

In the past, LM required a lot of manpower, computing resources, and time to achieve a task. Also, recently two-stage transfer learning is extremely popular in the NLP field. Therefore, BERT is one of the models that solve above things and use two-stage transfer learning. First, we pretraining LM model that have the advantage to let the model has a certain understand of natural language. Then use the model for feature extraction or fine tune downstream tasks by supervised approaches.

Let BERT perform two tasks at the same time when pre-training BERT:

1. Masked Language Model(MLM) [17]
   BERT masking once during data preprocessing. Google chooses 15% of token positions at random for prediction and 80% will use [MASK] token, 10% use a random token to replace and 10% will unchanged the original token. Then BERT need to predict the original token which is replace as above.

DOI:10.6814/NCCU202001728

2. Next Sentence Prediction(NSP)

   In order to let the model understand whether there is a relationship between sentences, author pre-train for a binarized NSP task that the model can feel more intuitive about learning.The specific method is to select sentence A and sentence B by pre-training corpus, 50% of the sentence after sentence A are correct sentences B,and the remaining 50% are taken randomly selected sentences from the corpus.

We show pre-training procedures for BERT as Figure 3.2. [CLS] is a special symbol token that added in begin of every input sequence, and [SEP] is a symbol token that separating sentences A and sentences B.[MASK] is only use at the pre-training that mask the word of sentence and predict the token.[UNK] role that does not appear in the BERT dictionary will replaced by this token.[PAD] is a zero padding mask token that will complete input sequences of different lengths to facilitate batch operations. The original BERT released two model sizes BERT-base (L=12, H=768, A=12 Total Parameters=110M) and BERT-Large(L=24, H=1024, A=16, Total Parameters=340M)

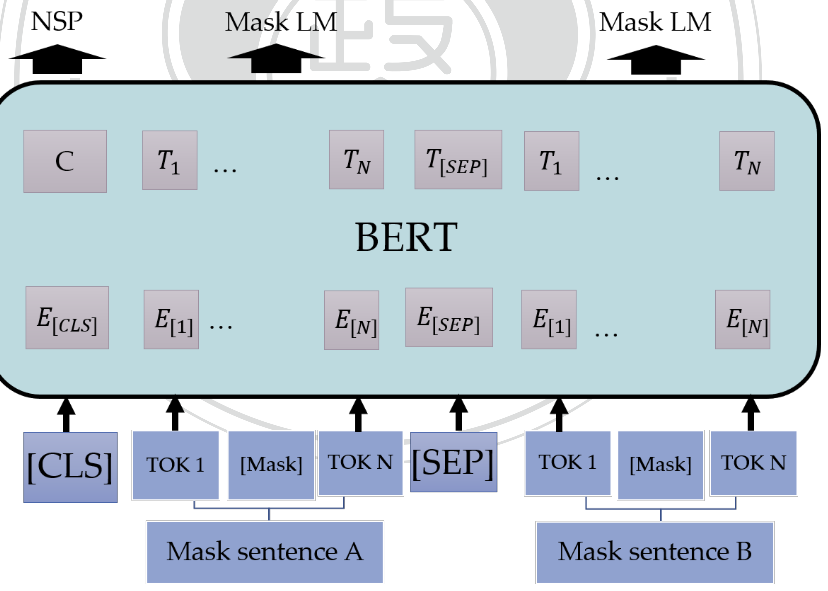


Figure 3.2: pre-training of BERT

Now, we will introduce the second-stage of transfer learning, with the first-stage pre-training model, we can use it for downstream tasks and have a good effect. Ther are 5 simple steps to fine tune BERT to solve new downsteam tasks. First, prepare raw text data. Second, convert raw text to BERT compatible input format. Third, add a new layer on BERT as a

DOI:10.6814/NCCU202001728

downstream task model then start train the downstream task model.Finally make predictions for new samples.

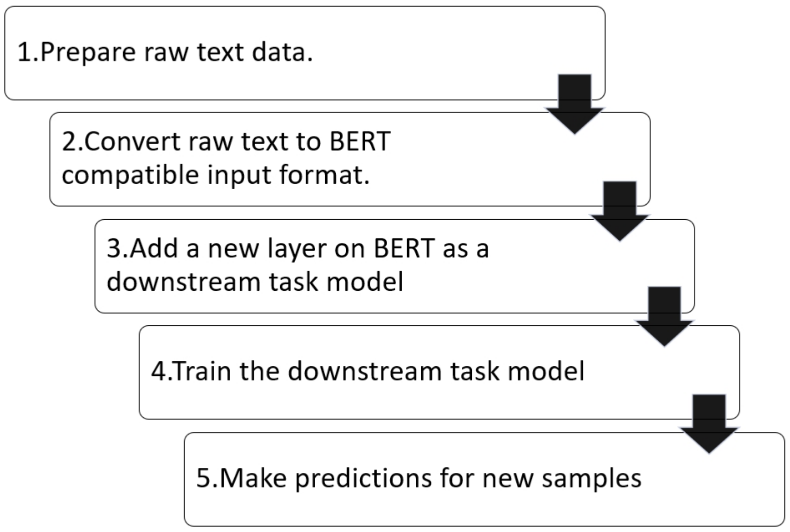

1.Prepare raw text data.
2.Convert raw text to BERT compatible input format.
3.Add a new layer on BERT as a downstream task model
4.Train the downstream task model
5.Make predictions for new samples

Figure 3.3: 5 steps of fine-tuning process

We extracted the 4 kinds of fine-tuning BERT to model the downstream task. Each case has a corresponding input method and output method. At the input, it can divided into single text and text pair. Single text input can be mainly applied in Single Sentence Tagging Tasks which let model identify the attributes of each word in the sentence and Single Sentence Classification Tasks which let model can identify property of the sentence. Text pairs applied in Sentence Pair Classification Task that model can recognize any relation between sentence1 and sentence2 and Question Answering Tasks allow the model to answer the question but this the answer is that it originally appeared in the text trained in fine-tune. In this paper, we only introduce the Single Sentence Classification Tasks that the fine-tune model structure is shown in Figure 3.4, its input is single sentence that we use $(x_1, x_2, x_3, x_4, ...)$ to represent it and output is a class. We will connect a linear classifier layer to the output token(the yellow columnar) and use the target function of the downstream task to train the classifier from the beginning and fine-tune the parameters of BERT.

DOI:10.6814/NCCU202001728

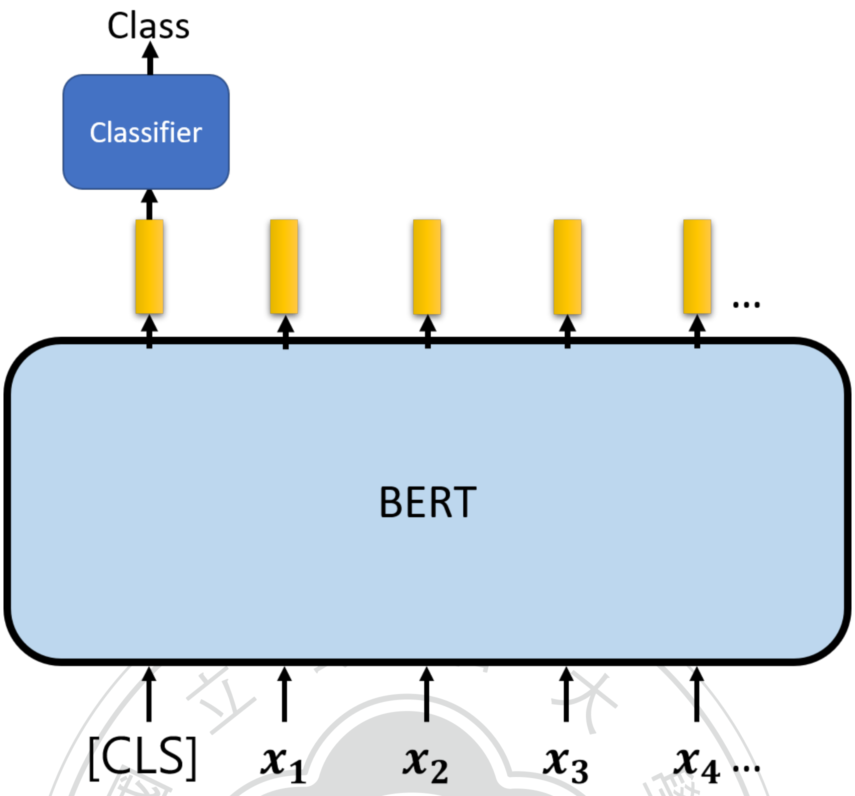


Figure 3.4: Single Sentence Classification Tasks

Compared with pre-training, the resource cost of fine-tuneing is relatively less. The time spent on each task takes only a few hours on the GPU, which is also the benefit of BERT.

3.3 RoBERTa

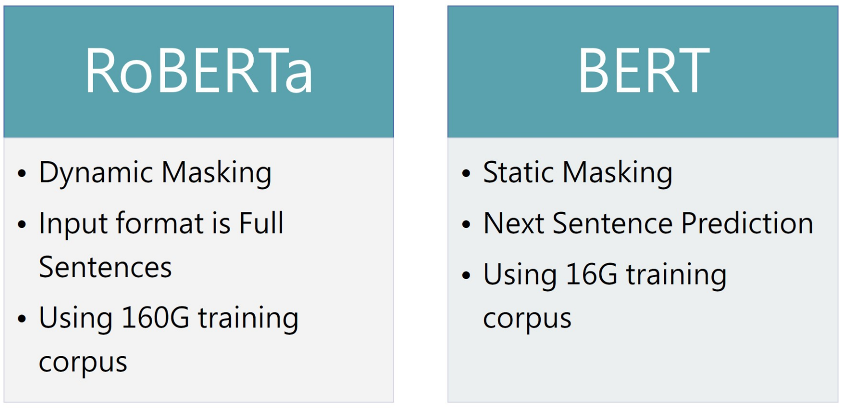
We previously introduced the power of BERT, but some people think that the original BERT is undertrained, so various optimization methods are adopted to improve the efficiency of BERT. [14]Each training is computationally expensive, so it also limits the number of parameters we adjust, and it also limits our understanding of how much progress can be made in the proposed model. Therefore, we can improve the process of training BERT from several aspects. The method include: (1) More rigorous research on the impact of hyperparameters; (2) improve training time to make time change longer; (3) increasing training batch size; (4) only training on longer sequences ; (5) use another masking pattern that is dynamic applied to the training data; (6) use larger training data(CC-NEWS), these methods have significant performance improvements.

DOI:10.6814/NCCU202001728

The method of pre-training tasks is also different from the static mask language model and next sentence prediction used by BERT before as discussed in Section 3.2. The task used by ROBERTA is dynamic mask language model and next sentence prediction task is removed. We illustrate the process of dynamic masking, RoBERTa in order to avoid the mask token has been repeated to cover the same training words in every epoch. Dynamic masking is to generate the masking pattern before starting model training (not during data preprocessing). The implementation show that the dynamic masking is more slightly better than static masking.

And another experiment showed that removing the next sentence prediction task and replacing it with the input format is Full-Sentences that contiguously from one or more documents that the total length is at most 512 tokens. Such changes slightly improve the effectiveness of downstream task.

We made a simple comparison table to compare BERT an RoBerta shown as Figure 3.5



RoBERTa	BERT
• Dynamic Masking • Input format is Full Sentences • Using 160G training corpus	• Static Masking • Next Sentence Prediction • Using 16G training corpus

Figure 3.5: Comparison Table of BERT and RoBERTa

3.4 Unsupervised Cross-lingual Representation Learning at Scale: XLM-R

Many language models now prove the effectiveness of pre-training methods for English natural language understanding. Therefore, these tasks is applied to cross-language and shows the effectiveness of cross-language pre-training. Although BERT also has a version that uses multiple languages for pre-training, the effect is not very good, so another pre-training method for cross-language is designed. Initialize BERT as a model for unsupervised machine translation. The model is called XLM (Lample and Conneau, 2019) [12], XLM provides three kinds of

DOI:10.6814/NCCU202001728

pre-training task, the first task they called it Causal Language Model(CLM) that is to predict probability of the next word for a given sentence but this technique cannot be extended to cross languages, so XLM only keep the first word in each batch and regardless of context. The second task is use Mask Language Model that is same as BERT but have little bit difference to BERT, XLM use an arbitrary number of sentences instead of pairs of sentences. In oreder to balance the inequality of tokese such as punctuation and stop words.

Third, we are going to introduce how the author can transform BERT in the following ways, In BERT, each sample is constructed in one language for each training. XLM's improvement to it is that each training sample contains the same text in both languages. Like BERT, the goal of this model is to predict the word masked in a sentence, but with a new architecture, the model can use the context of one language to predict words in another language. This method is called Translation Language Model(TLM) that is a supervised method to pre-train and TLM is an extension of MLM, for example, we have a pair of sentences in Chinese and English, the Chinese sentence is "我是學生" and the English sentence is "I am a student". Then "學生" is the masked word in chinese sentence and can use "a student" to prdecit it, and "am" is the masked word in english sentence and also use "是" to predict it, we show it as in Figure3.6.

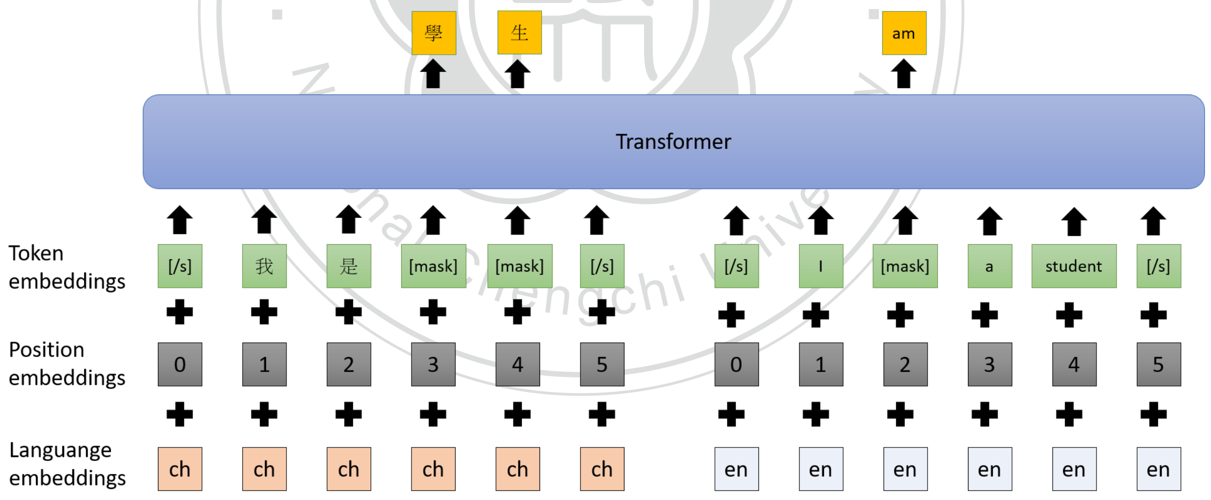


Figure 3.6: Translatoin Language Modeling(TLM)

The above model provides good cross-language pre-training, but when training multi-language models are use Wikipedia data set, and the ability to represent useful resources in low-resource languages is still limited. Therefore, by simply increasing the model, the problem of low resource languages can be effectively alleviated. Specifically, more than 2TB of CommonCrawl data sets that have been preprocessed in 100 languages are used to train cross-

DOI:10.6814/NCCU202001728

lingual representations in a self-supervised manner. This includes generating new unlabeled corpora for low-resource languages and expanding the amount of training data used in these languages by 2 orders of magnitude. They called it XLM-R.(Conneau et al., 2019) [4] And difference to Lample and Conneau, 2019 [12], XLM-R do not use language embeddings in pre-training that makes XLM-R can handle code switching better. XLM-R uses the above tasks and expands the pre-training data set to allow it have good performance in cross-lingual beckmark.

國立政治大學
National Chengchi University

DOI:10.6814/NCCU202001728

Chapter 4

Cross-language prediction and Imbalanced Data

In this paper, we want to use deep learning to predict whether a sentence is a toxic message . but we want to use english training data set to predict the test set of chinese sentences,and will compared with the Chinese and English training data set. The Englsih data set we collect is public information on Jigsaw Multilingual Toxic Comment Classification of competition of Kaggle, and for the Chinese data set we collected Hate board in Taiwan's well-known social media PTT.

4.1 Toxic Message

In real life, social media is inseparable, according to statistics, everyone spends at least 1.5 hours on social media every day, and there are toxic comment or messages in conversations on the internet, where toxicity is defined as anything rude, uncomfortable or otherwise likely to make someone feel insulted, leads to a poor user experience and effects the mood. If toxic messages can be identified in other languages and can be used and contributed in various languages, we will have a safer and more collaborative Internet.

DOI:10.6814/NCCU202001728

4.2 Imbalanced Data

Usually, the toxic comment identified can look upon it as a binary classification problem which consisted of one positive group and one negative group i.e. toxic comment and general comment. When collecting data, there are far fewer toxic comment than general comment. In this section, we will discuss the imbalanced data in classification task and explantion the problem in this paper.

Imbalanced data occurs in many different applications, and the frequency of these data in one of the categories is extremely low, such as computer security [3], disease diagnosis [15], image recognition [11]. Therefore, we want to judge the imbalanced level of dataset and the below formula [1] can be represents the imbalance ratio of dataset, in the other words, the ratio of maximum class size and minimum class size.

$$\rho = \frac{\max_i(|D_i|)}{\min_j(|D_j|)}$$

where D_i is a set of samples in class i. For example, if our training dataset's largest class which has 1000 samples and smallest class which has 100 samples, then the imbalance ratio $\rho = 10$.

In this chapter, we will discuss the methods that can be used when encountering unbalanced data situations.

4.2.1 Data-level methods

In this section, we achieve all classes are balanced by adjusting the number of samples of different classes, the following methods can be used regardless of binary classification or multiple classification.

1. Random-oversampling(ROS): This method is called oversampling which does not perform any processing on the majority class samples, but increases the number of minority class samples to improve the training of the minority. [7] This method uses random copy to balance the two types of data, which is easy to implement, but due to repeated copying of minority samples, the possibility of overfitting of the classification algorithm is increased. For example, we use this method to apply to our Chinese training data set as Figure 4.3.

DOI:10.6814/NCCU202001728

國立政治大學 National Chengchi University

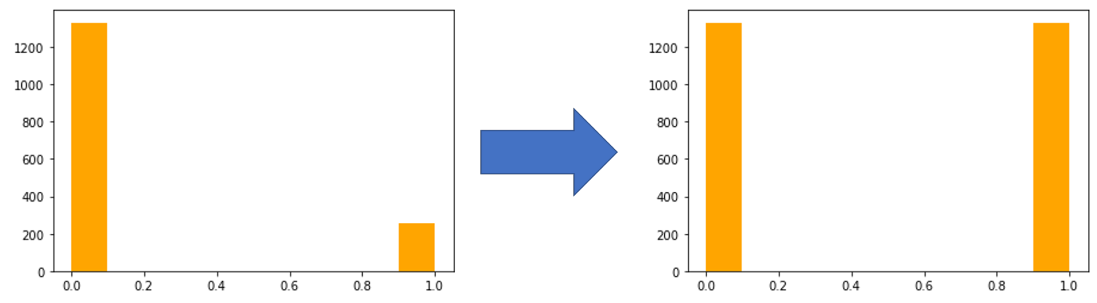


Figure 4.1: Random oversampling(ROS)

2. Random-undersampling(RUS):

   Undersampling is another common solution of imbalanced data [7], Contrary to the oversampling method, the undersampling method improves the classification accuracy of the minority class by reducing the number of samples of the majority class. Random undersampling is one of the simplest undersampling methods, by randomly selecting a part of the majority class samples to achieve a balanced sample. For the purpose of quantity, the training time of the model can be effectively shortened. However, randomly discarding samples may remove potentially useful information in largest classes, thereby reducing the performance of the model to identify sample. We also get example, we use undersampling method to apply to our Chinese training data set as Figure 4.4.

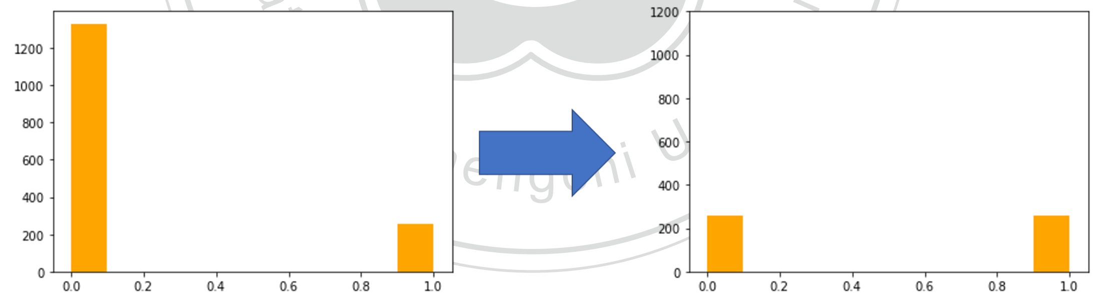


Figure 4.2: Random undersampling(RUS)

In data-level methods, we usually use ROS to eliminate class imbalance and have good impact on small data of class. But if our data set is much bigger or have extreme class imbalance i.e. ρ is large, performance of oversampling is maybe not good because it will increase too many repeated samples led to overfitting and increase the time of training. On the other hand, we believe that RUS can obtain better performance on large imbalanced data set since it can reduce the time of training.

DOI:10.6814/NCCU202001728

Chapter 5

Experiments

In this chapter, We used the language model ”xlm-roberta” mentioned earlier to predict whether the comment is a toxic message so our classes is simply divided into binary classifications that are true class and false class. As we mentioned in Chapter 4, when the comment is the toxic message we classified in the true class and it is represented by 1 and if the comment is not the toxic message we classified in the false class and it is represented by 0.

Our English training dataset's (ENtrain) largest class 0 is represent not toxicity which has 200,000 samples and smallest class 1 is represent toxicity which has 20,000 samples as the Figure 5.1, then the imbalance ratio $\rho = 10$, and the Chinese training dataset's (CHtrain) largest class 0 is represent not toxicity has 1,331 samples and smallest class 1 is represent toxicity has 258 samples as the Figure 5.2, then the imbalance ratio is about $\rho = 5.16$. And the testing dataset (CHtest) is Chinese and has a total of 400 data and their distribution is shown in the figure5.3.

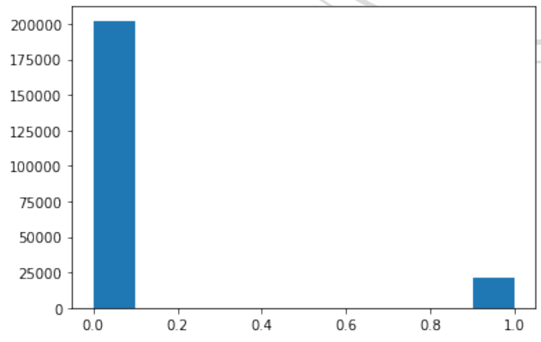


Figure 5.1: training dataset of English

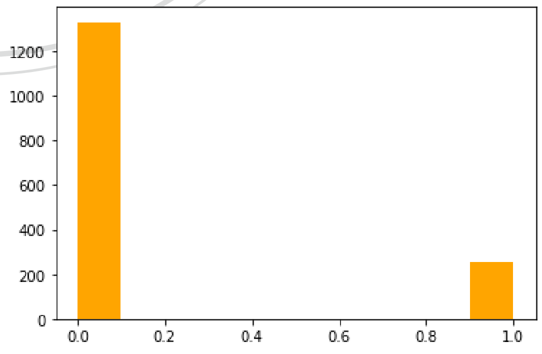


Figure 5.2: training dataset of Chinese

DOI:10.6814/NCCU202001728

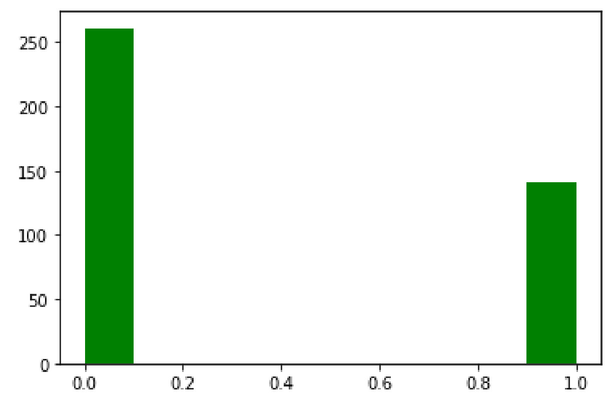


Figure 5.3: testing dataset of Chinese

5.1 Prediction model

After many experiments, we fixed the following parameters during fine-tuning, we set the learning rate $\eta = 4e-5$ when we train Chinese and the learning rate $\eta = 3e-5$ in English, batch size of train = 4 that it means the numbers of samples we input the model. We repeat the training five times, so we set epoch = 5 and set the maximum length of the sentence to 328, which means that words beyond the 328th will be ignored. At the first place we put the [CLS] token and in the down stream we put the classifier model at the output of [CLS] token then we objective is train the classifier model, we input the monolingual comments then classify the comments is whether a toxic comment if the comment is toxicity we output is 1(True) and opposite is 0(False), we named this model is Toxic Comments Detection Model (TCDM).

5.2 English training dataset

We use the above training dataset of English and training parameters without any changes to the data then we name this model M_o, and the accuracy score of the prediction CHtrain and CHtest is as follows Figure 5.4. The class of 1(True) and class of 0(False) in the Figure5.4 are all from CHtest.

	CHtrain(avg)	CHtest(avg)	1(True)	0(False)
M_o	89.7%	79.1%	44.7%	97.7%

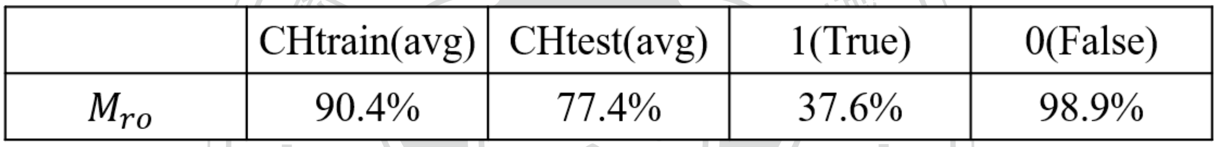
Figure 5.4: Average accuracy of M_o

DOI:10.6814/NCCU202001728

According to Figure 5.4, we can observe that the prediction of class 0 is more accurate. Sine the class 0 is relatively large, the average accuracy is also relatively better. To improve the M_o, we will use random-oversampling method and random-undersampling method that observe whether the accuracy has improved.

5.2.1 Random-oversampling(ROS)

In this section, we will adjust the smaller class(1) in ENtrain as described in section 4.2.1. We will randomly select samples from the smaller class(1) and copy them until the number of samples is the same as the large class(0). This method makes our data become balanced data. We use the above parameters and ROS method and name this model M_{ro}. We can see that it achieves an accuracy of 98.9% in class 0 but has a worse performance than M_o in class 1. That's because the sample of class 0 in ENtrain is more large than class 1, which cause the model M_{ro} is overfitting on class 1.

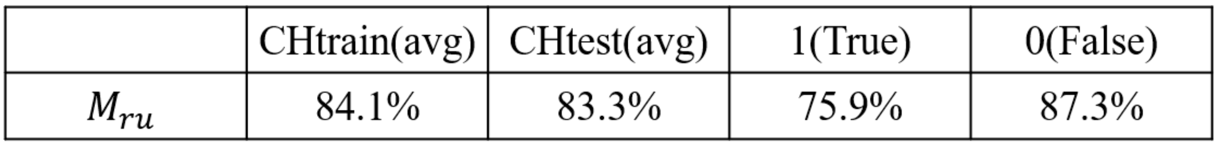


	CHtrain(avg)	CHtest(avg)	1(True)	0(False)
M_{ro}	90.4%	77.4%	37.6%	98.9%

Figure 5.5: Average accuracy of M_o and M_{ro}

5.2.2 Random-undersampling(RUS)

Compared to section 5.2.1, we use undersampling in this section, we remove the sample of larger class(0) to the same number as the sample of smaller class(1). After the data goes through this method, the structure of model use the same parameters as above and name it M_{ru}. The M_{ru} model has better performance in class 1 and CHtest, espeially in class 1, which has a great improvement.



	CHtrain(avg)	CHtest(avg)	1(True)	0(False)
M_{ru}	84.1%	83.3%	75.9%	87.3%

Figure 5.6: Accuracy of M_{ru}

And why the bad performance in CHtrain is because they have more sample of class 0. We

立 政 治 大 National Chengchi University

DOI:10.6814/NCCU202001728

think that because some important samples in the class 0 of ENtrain may be deleted, which leads to the bad performance of the RUS method. Then our model M_{ru} can not capture the feature of sentence.

5.3 Chinese training dataset

We use CHtrain in this section and will introduce the same method to fine-tuning the model. In our many experiments, we found that Chinese have good performance when learning rate set at $\eta = 4e - 5$, which is better than at $\eta = 3e - 5$. Therefore, we use the same parameter as section 5.1 that we mentioned. We name the model called M_c that is only use CHtrain data and not use any method. The performance as Figuer 5.7, we can see the accuracy score of class 0 is 95.4% that is also better than class 1.

	CHtrain(avg)	CHtest(avg)	1(True)	0(False)
M_c	92.7%	85%	65.2%	95.8%

Figure 5.7: Average accuracy of M_c

5.3.1 Random-oversampling(ROS)

In this section, we will adjust the sample of class 1 from CHtrain, let it random copy from 258 samples to 1331 samples. The structure of model is the same parameter. We name this model is M_{cro} and it accuracy score as Figure 5.8. M_{cro} have a good perfermance at class 0 that accuracy score is 98.9%. But still have bad performance at class 1 like section 5.2.1, compare the accuracy score at class 1 of M_c is less than 7%. The reason is model that overfitting on class 1, then cause the model learn class 1 not well.

	CHtrain(avg)	CHtest(avg)	1(True)	0(False)
M_{cro}	99.8%	84.5%	58.2%	98.9%

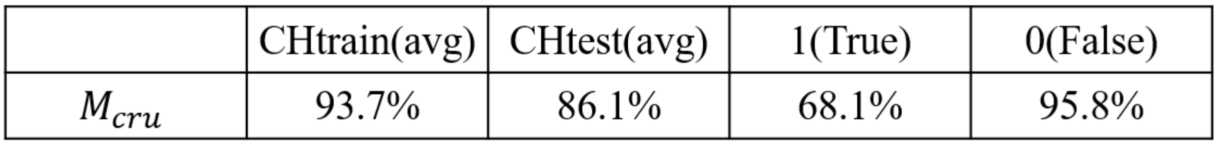
Figure 5.8: Average accuracy of M_{cro}

政 治 National Chengchi University

DOI:10.6814/NCCU202001728

5.3.2 Random-undersampling(RUS)

We also use the undersampling method on CHtrain data set. We will delete the sample of class 0 by randomly that until to 258 samples. We call this model is M_{cru} and the performance shown as Figure 5.9. It performed better than the original model M_c in every task. The model M_{cru} achieve to accuracy score at 86.1% in CHtest that an increase of 1.3%.

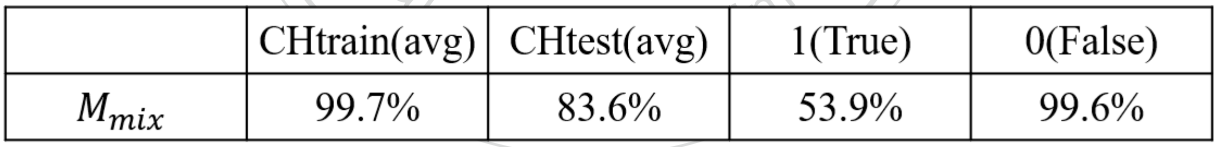


	CHtrain(avg)	CHtest(avg)	1(True)	0(False)
M_{cru}	93.7%	86.1%	68.1%	95.8%

Figure 5.9: Average accuracy of M_{cru}

5.4 Combined training dataset

In this section, we combined our training data set which mean we combine the Entrain and CHtrain. In our experiment, we attempt to adjust the ratio of two language samples of training set. We set the learning rate difference to above that is $\eta = 3.5e - 5$ and other parameters are the same. We fine tune four kind of model to predict the Chtest and observe the performance of models. The first model, we do not make any changes to directly merge the Entrain and Chtrain and called this model M_{mix}. The accuracy score shown as Figure 5.10 and it have good performance on CHtrain is 99.7%.



	CHtrain(avg)	CHtest(avg)	1(True)	0(False)
M_{mix}	99.7%	83.6%	53.9%	99.6%

Figure 5.10: Average accuracy of M_{mix}

5.4.1 Random-oversampling(ROS)

At the beginning in this chapter, we mentioned the number of data of Chtrain and Entrain are difference, in particular the number of class 1 of Entrain still exceed largest class of CHtrain. Hence we adjust the class 0 and class 1 of ENtrain to 1300 it same as number of CHtrain class 0. Also we random choose class 1 of CHrain to increase the data to same as number of CHtrain class 0. We name this model M_{mro} and show it performance at Figure 5.11, then compare to

國立政治大學 National Chengchi University

DOI:10.6814/NCCU202001728

model M_{cro} have significant improvement in performance in all aspects. The accuracy score of CHtest have 87.1%.

	CHtrain(avg)	CHtest(avg)	1(True)	0(False)
M_{mro}	96.2%	87.1%	66.7%	98.1%

Figure 5.11: Average accuracy of M_{mru}

5.4.2 Random-undersampling(RUS)

We remove the numbers of class 0 and class 1 of ENtrain to same as class 1 of Chtrain and so do class 0 of CHtrain that mean all numbers of class are 250 then use it to fine tune model. This model is name M_{mru} and shown it performance at Figure 5.12.

	CHtrain(avg)	CHtest(avg)	1(True)	0(False)
M_{mru}	92.5%	84.1%	63.1%	95.4%

Figure 5.12: Average accuracy of M_{mru}

5.4.3 Adjust ratio of sample

We find if we adjust the class 1 and class 0 of Entrain to same as ratio of class of Chtrain which mean the ratio of class 1 and class 0 is $5:1$ then we get best performance shown as Figure 5.13. We use the same parameter and name this model M_{msp}. We can observe M_{mix} compare the above model that have the best performance on CHtest and the accuracy score is 88.3%. It also have the good performance at the class 0 and the accuracy score is 71.6%.

	CHtrain(avg)	CHtest(avg)	1(True)	0(False)
M_{msp}	95.3%	88.3%	71.6%	97.8%

Figure 5.13: Average accuracy of M_{msp}

DOI:10.6814/NCCU202001728

Chapter 6

Conclusion

In this paper, we want to research the feasibility that use the language different from target language which to predict the target language. Because English is high resource language in many fields that is easier to find data than other languages. Therefore, we use the English training dataset to predict the Chinese testing set. This problem is in cross-lingual field, so we use the cross-lingual model (XLM-RoBerta base version), and collect data from the Jigsaw Multilingual Toxic Comment Classification on Kaggle and Hate board in social media PTT. As the following Figure 6.1 is all model compare list. Then we can observe the model just only use the English training dataset to fine-tuning model that have good performance, especially on class 1.

model	CHtrain(avg)	CHtest(avg)	1(True)	0(False)
M_o	89.7%	79.1%	44.7%	97.7%
M_{ro}	90.4%	77.4%	37.6%	98.9%
M_{ru}	84.1%	83.4%	**75.9%**	88.5%
M_c	92.7%	84.8%	65.2%	95.4%
M_{cro}	**99.8%**	84.6%	58.2%	98.9%
M_{cru}	93.7%	86.1%	68.1%	95.8%
M_{mix}	99.7%	83.6%	53.9%	**99.6%**
M_{mro}	96.2%	87.1%	66.7%	98.1%
M_{mru}	92.5%	84.1%	63.1%	95.4%
M_{msp}	95.3%	**88.3%**	71.6%	97.8%

Figure 6.1: Average accuracy of all model

Since the English training set is more larger then Chinese training set such that we can only

DOI:10.6814/NCCU202001728

use English training set to get the best performence on class 1. And we get the good performance when we combine the English and Chinese training data set be as new training set. On the other hands, when our target language has a low resource in any field, we can obtain data from other languages that have high resource in this fields, then combine two languages data or only use high resource language data to fine-tune the model. At the follow, there have serval directions we can discuss in this problem at the future:

1. The ratio of the combined training set can have a perfect ratio to increase the accuracy.
2. The structure of cross-lingual model can be adjust it pre-training method to improve score of task.
3. Try to use other methods that can solve unbalanced data that is language.

國立政治大學 · National Chengchi University ·

DOI:10.6814/NCCU202001728

Bibliography

[1] Mateusz Buda, Atsuto Maki, and Maciej A Mazurowski. A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks*, 106:249–259, 2018.

[2] KR Chowdhary. Natural language processing. In *Fundamentals of Artificial Intelligence*, pages 603–649. Springer, 2020.

[3] David A Cieslak, Nitesh V Chawla, and Aaron Striegel. Combating imbalance in network intrusion datasets. In *GrC*, pages 732–737, 2006.

[4] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised cross-lingual representation learning at scale, 2019.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018.

[6] Chris Drummond, Robert C Holte, et al. C4. 5, class imbalance, and cost sensitivity: why under-sampling beats over-sampling. In *Workshop on learning from imbalanced datasets II*, volume 11, pages 1–8. Citeseer, 2003.

[7] Guo Haixiang, Li Yijing, Jennifer Shang, Gu Mingyun, Huang Yuanyue, and Gong Bing. Learning from class-imbalanced data: Review of methods and applications. *Expert Systems with Applications*, 73:220–239, 2017.

[8] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

國立政治大學 National Chengchi University

DOI:10.6814/NCCU202001728

[9] Hossein Hosseini, Sreeram Kannan, Baosen Zhang, and Radha Poovendran. Deceiving google's perspective api built for detecting toxic comments, 2017.

[10] Anil K Jain, Jianchang Mao, and KM Mohiuddin. Artificial neural networks: A tutorial. *Computer*, (3):31–44, 1996.

[11] Miroslav Kubat, Robert C Holte, and Stan Matwin. Machine learning for the detection of oil spills in satellite radar images. *Machine learning*, 30(2-3):195–215, 1998.

[12] Guillaume Lample and Alexis Conneau. Cross-lingual language model pretraining. *CoRR*, abs/1901.07291, 2019.

[13] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553): 436–444, 2015.

[14] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[15] R Bharat Rao, Sriram Krishnan, and Radu Stefan Niculescu. Data mining for improved cardiac care. *ACM SIGKDD Explorations Newsletter*, 8(1):3–10, 2006.

[16] Daniel Svozil, Vladimir Kvasnicka, and Jiri Pospichal. Introduction to multi-layer feed-forward neural networks. *Chemometrics and intelligent laboratory systems*, 39(1):43–62, 1997.

[17] Wilson L. Taylor. "cloze procedure": A new tool for measuring readability. *Journalism Quarterly*, 30(4):415–433, 1953.

[18] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc., 2017.

[19] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva

國立政治大學 National Chengchi University

DOI:10.6814/NCCU202001728

Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation, 2016.

[20] Show-Jane Yen and Yue-Shi Lee. Under-sampling approaches for improving prediction of the minority class in an imbalanced dataset. In *Intelligent Control and Automation*, pages 731–740. Springer, 2006.

[21] Dong Yu and Li Deng. *Automatic Speech Recognition: A Deep Learning Approach*. Springer Publishing Company, Incorporated, 2014.

國立政治大學 · National Chengchi University ·

DOI:10.6814/NCCU202001728