Report of Deep Learning for Natural Langauge Processing 4

Text generation using Seq2seq and transformer models

Jiayi Zhang
zy2303814@buaa.edu.cn

Abstract

In this experiment, two different models of Seq2Seq and Transformer are trained to realize text generation by using the given corpus, and the advantages and disadvantages of the two methods are discussed and compared. After comparison, it is concluded that the transformer can train faster than the normal seq2seq model when the training effect is similar.

Introduction

Text generation is a key task in NLP, which aims to generate coherent, accurate, and natural text based on input information. With the development of deep learning technology, Seq2Seq models and Transformer models have achieved remarkable success in the field of text generation.[1]

In 2014, Cho et al. first proposed the Seq2Seq (sequence-to-sequence) model in recurrent neural networks (RNN). Compared with the traditional statistical translation model, the Seq2Seq model greatly simplifies the processing flow of sequence conversion tasks.The Seq2Seq model is a sequence-to-sequence encoder-decoder structure consisting of an encoder and a decoder. As is shown in Figure 1, The encoder encodes an input sequence (such as source language text) into a fixed-length vector, and the decoder decodes this vector into a target sequence (such as target language text).

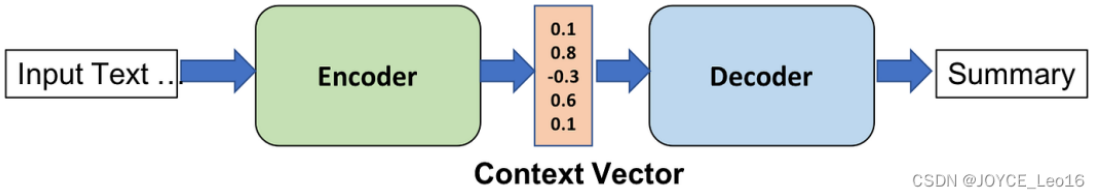
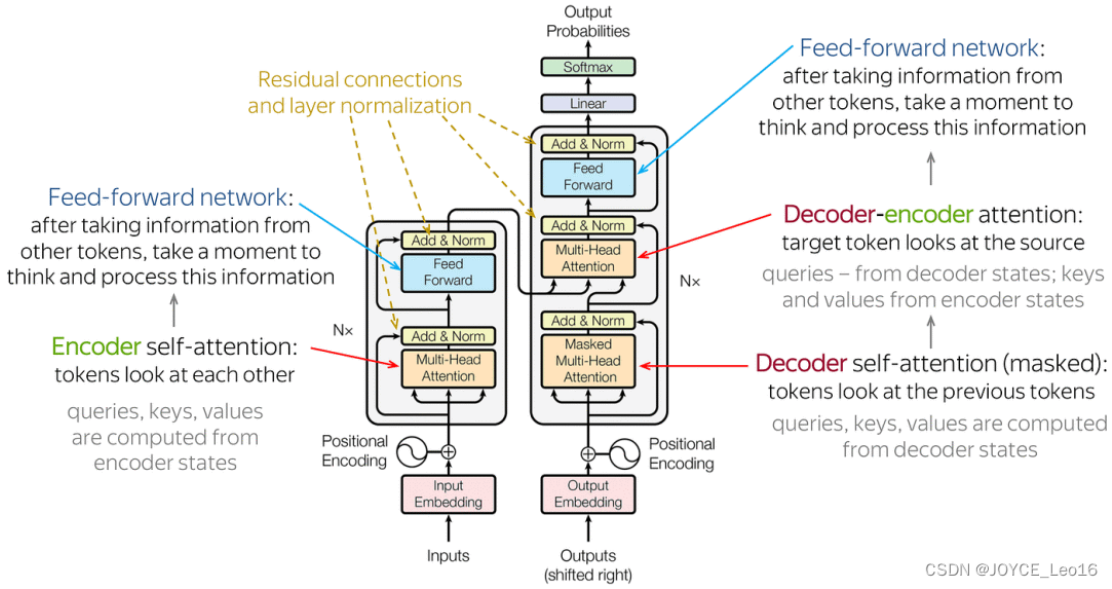


Figure 1 Seq2seq Model[2]

The Transformer model was introduced in the paper "Attention Is All You Need" by Ashish Vaswani, Noam Shazeer et al. This groundbreaking work was presented at the 2017 Conference on Neural Information Processing Systems (NeurIPS). The core innovation of the Transformer model is the self-attention mechanism, which allows the model to weigh the importance of different words in a sequence relative to each other. This mechanism enables the model to capture dependencies between words regardless of their position in the sequence.

As is shown in Figure 2, Similar to traditional Seq2Seq models, Transformers have an encoder-decoder structure. However, unlike RNN-based models, both the encoder and decoder in a

Transformer are composed of multiple identical layers, each containing two main components: a



multi-head self-attention mechanism and a position-wise fully connected feed-forward network.

Figure 2 Transformer Mode

Since Transformers do not use recurrence or convolution, they incorporate positional encoding to provide information about the position of words in the sequence. This encoding is added to the input embeddings to retain the order of the sequence.

Each self-attention layer consists of multiple heads that allow the model to focus on different parts of the sequence simultaneously. This multi-head attention mechanism helps the model capture various aspects of the relationships between words.

After the self-attention mechanism, each layer includes a fully connected feed-forward network applied independently to each position. This network consists of two linear transformations with a ReLU activation in between.

Each sub-layer (self-attention and feed-forward network) in the encoder and decoder is followed by a layer normalization and a residual connection, which helps in stabilizing and accelerating the training process.

Since its introduction, the Transformer model has become the foundation for many state-of-the-art models in natural language processing (NLP), including BERT, GPT, and T5. It is widely used in tasks such as machine translation, text summarization, and language modeling.

Methodology

M1: Seq2Seq Model

The Seq2Seq model is a sequence-to-sequence encoder-decoder structure consisting of an encoder and a decoder. The encoder encodes an input sequence (such as source language text) into a fixed-length vector, and the decoder decodes this vector into a target sequence (such as target language text). Seq2Seq model mainly includes the following components[1]:

- Vocabulary: Maps words to a unique integer index.
- Encoder (Encoder) : RNN(recurrent neural network) or LSTM(long short-term memory network) are usually used to process input sequences and generate hidden states.
- Decoder: Use RNNS or LSTMS to generate target sequences by continuously predicting the next word.

- Attention mechanism: Improve the predictive power of the decoder so that it can focus on certain time steps of the encoder.

The encode (Figure 3) is the part of the Seq2Seq model that converts the input sequence into a fixed-length context vector. It uses recurrent neural networks (RNNS) or their variants (e.g. LSTM, GRU) to implement this conversion process. During the encoding process, the encoder reads the elements in the input sequence one by one and updates their internal hidden state. After the encoding is complete, the encoder passes the final hidden state or some transformed hidden state to the decoder as a context vector.

The decoder (Figure 3) is another part of the Seq2Seq model that is responsible for generating output sequences from context vectors. It also uses recurrent neural networks (RNNS) or their variants (e.g. LSTM, GRU) to implement the generation process. At each time step, the decoder generates the output of the current time step based on the output of the previous time step, the current hidden state, and the context vector. The decoder completes the task of generating the entire sequence by gradually generating each element in the output sequence.

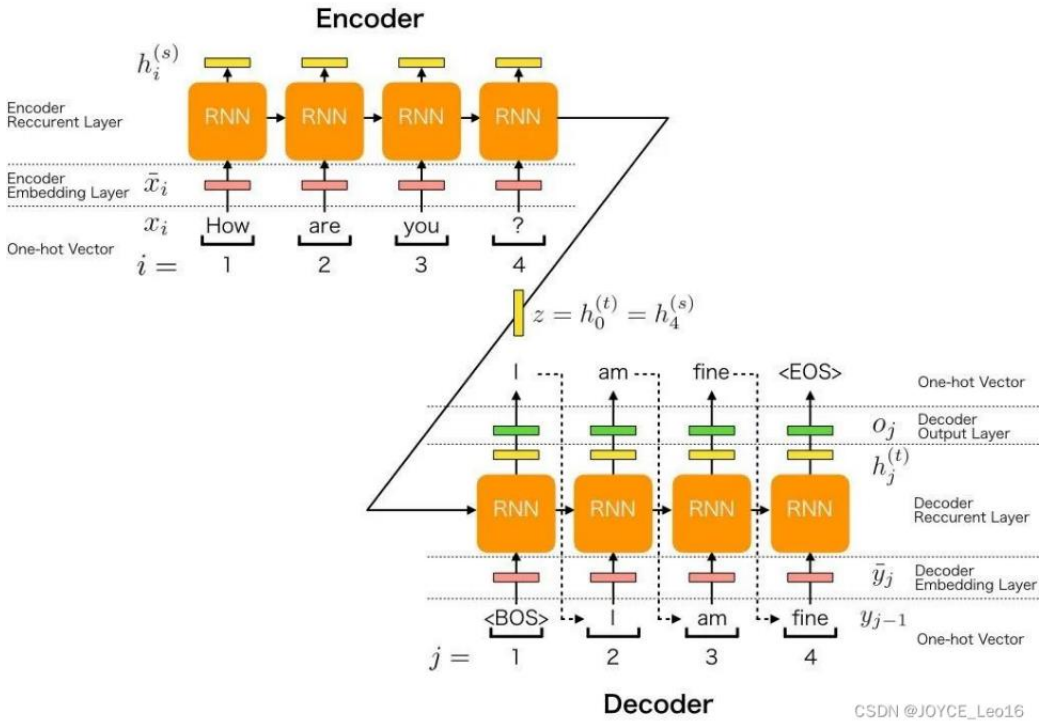


Figure 3 Seq2seq Model[2]

Because traditional machine translation does not work well on long sentences, it is difficult for fixed-length vectors to contain all the semantic details of the sentence. The core idea of the attention mechanism is that at the time of generating each output word, the model is able to focus on the relevant part of the input sequence. Its core logic is to go from focusing on the whole to focusing on the main points. When the Attention mechanism processes long text, it can capture important points without losing important information.The attention mechanism was originally introduced to address the performance degradation of long sentences (over 50 words) encountered in machine translation.

M2: Transformer Model

The Transformer model is a variant of the Seq2Seq model, and its main feature is that it is entirely based on self-attention mechanism and has no recursive structure. Its main components

include:

- Vocabulary: Maps words to a unique integer index.
- Encoder (Encoder) : Multiple self-attention heads are used to process input sequences, generating multiple context vectors.
- Decoder: The use of multiple self-attentional heads to generate a target sequence by continuously predicting the next word.
- Positional Encoding: To solve the problem of missing positional information in the Transformer model, positional information is added to the input vector.

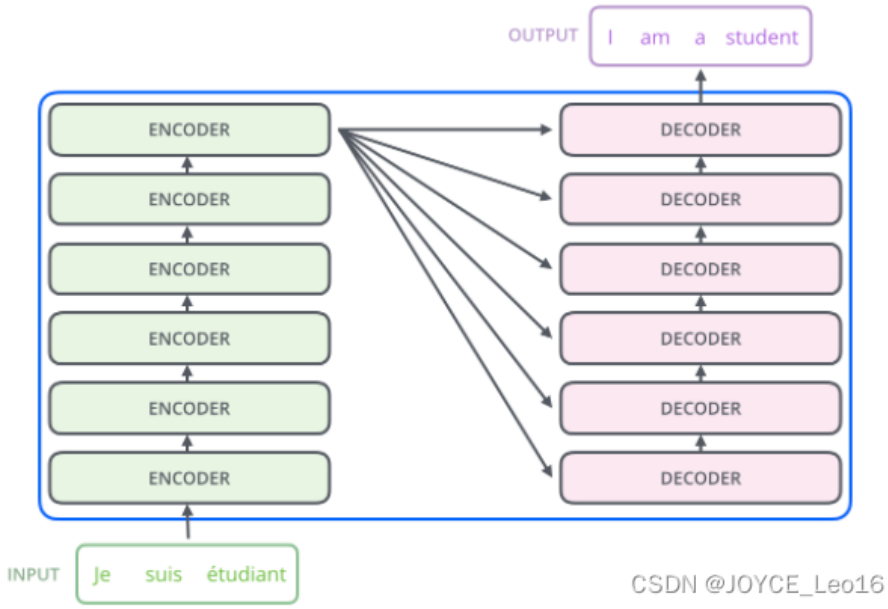


Figure 4 Transformer Model[2]

As shown in Figure 4 (N = 6), The encoder is composed of *N* identical layers. Each layer has two main sub-layers: a multi-head self-attention mechanism and a feed-forward neural network. The input to the encoder is a sequence of word embeddings with added positional encodings. (Figure 5)

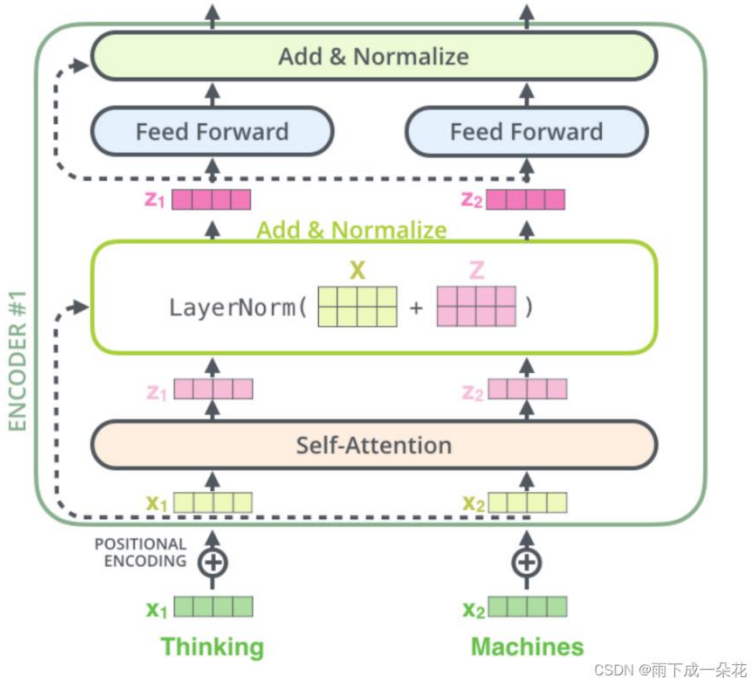


Figure 5 Encoder[2]

The decoder is also composed of N identical layers. In addition to the two sub-layers in each encoder layer, each decoder layer has a third sub-layer that performs multi-head attention over the output of the encoder stack. The decoder generates the output sequence one token at a time, using previously generated tokens to predict the next token. (Figure 6)

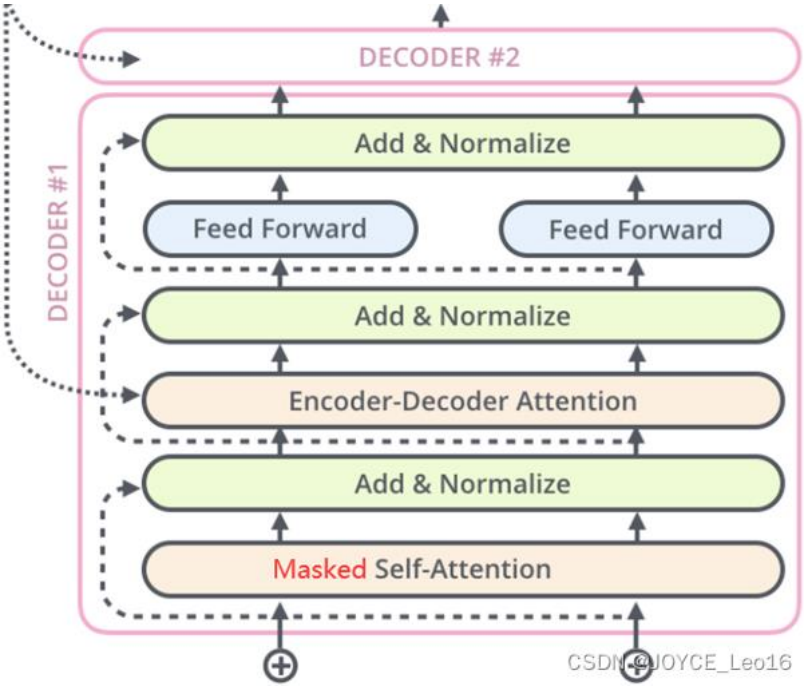


Figure 6 Decoder[2]

Experimental Studies

Step1: Data Preprocessing

1. **Initialize the List of Stop Words and Special Vocabulary**: The script begins by reading Chinese punctuation marks, stop words, character names, martial arts techniques, and sect names from files, adding them to corresponding lists. It also uses the Jieba segmentation tool to add these character names, martial arts techniques, and sect names to the vocabulary, ensuring they are correctly recognized during subsequent segmentation processes.

2. **Read Novel Names**: The script reads from a text file containing a list of novel names, storing these names in a list.

3. **Text Preprocessing and Segmentation**: For each novel in the list, the script opens the corresponding text file for reading. During the reading process, it first removes specific advertisement information and non-Chinese characters from the text, then uses Jieba for word segmentation. After segmentation, the script further filters out stop words and whitespace, saving the remaining words.

4. **Data Saving:** Finally, the script saves the segmented results and randomly selected paragraphs to a CSV file. Each list of segmented words is converted into sentence form and written as a row in the CSV file.

After processing, each sentence of the novel text is divided by a period after removing the stop word, and the format is as follows:

“白马啸西风。

得得得得得得。
得得得得得得。
在黄沙莽莽的回疆大漠之上尘沙飞起两丈来高两骑马一前一後的急。
驰而来前面是匹高腿长身的白马马上骑著个少妇怀中搂著个七八岁的。
小姑娘後面是匹枣红马马背上伏著的是个高瘦的汉子。
那汉子左边背心上却插著一枝长箭鲜血从他背心流到马背上又流到。
地下滴入了黄沙之中他不敢伸手拔箭只怕这枝箭一拔下来就会支持。
不住立时倒毙谁不死呢那也没什麽可是谁来照料前面的娇妻幼女。
在身後凶悍毒辣的敌人正在紧紧追踪。
…….(Close to 2000 lines of sentences)”

Step2: Train the Seq2Seq model

1. **Data preprocessing:** The data_pre function is responsible for reading the data file, cleaning the text using regular expressions, counting word frequency, and building a vocabulary. Then, the text is transformed into an index sequence, and the dataset and data loader are created.

2. **Initialize model parameters:** Create encoders, attention mechanisms, decoders, and seq2seq model instances.

・The training parameters are defined as:

・input and output dimensions (INPUT_DIM, OUTPUT_DIM) : vocabulary size.

・Encoder and decoder embedding dimensions (ENC_EMB_DIM, DEC_EMB_DIM) : both are 256.

・Hidden dimension (HID_DIM) : 256.

・Number of encoders and decoders (N_LAYERS) : 2.

・The dropout rate of encoder and decoder (ENC_DROPOUT, DEC_DROPOUT) : both are 0.1.

・Learning rate: 0.0002.

・Training cycles (N_EPOCHS) : 50.

・Gradient clipping threshold (CLIP) : 1.

3. **Model training:** Define the optimizer, loss function, and GradScaler. The model is trained for the specified number of cycles and training losses are recorded.

4. **Text generation:** The Seq2seq model is set to evaluation mode, and then the input sentences are converted into index sequences, encapsulated as tensors. The input sequence is processed using the encoder of the model, and the output, hidden state, and cell state of the encoder are obtained. Initializes the target sequence index, adding the sequence start character <sos> to the target index list. Loops generate each word until the maximum length is reached or the sequence terminator <eos> is generated. In each iteration, the previous predicted word is converted into a tensor, the decoder is fed, and the decoder's output is used to predict the next word. If the predicted word is a sequence terminator, the loop is terminated. Finally, the target index list is converted back to a list of words, with the start and end symbols excluded, and the words are then concatenated to form the resulting sentence.

The experimental results are shown below：

Model training output	
Epoch: 1/50, Train Loss: 7.4032, Time: 31.29 seconds	Epoch: 26/50, Train Loss: 2.2949, Time: 31.24 seconds
Epoch: 2/50, Train Loss: 6.4096, Time: 31.93 seconds	Epoch: 27/50, Train Loss: 2.0441, Time: 31.02 seconds
Epoch: 3/50, Train Loss: 6.1906, Time: 29.69 seconds	Epoch: 28/50, Train Loss: 1.8507, Time: 30.92 seconds

Epoch: 4/50, Train Loss: 6.0510, Time: 29.82 seconds	Epoch: 29/50, Train Loss: 1.7171, Time: 30.48 seconds
Epoch: 5/50, Train Loss: 5.8982, Time: 30.00 seconds	Epoch: 30/50, Train Loss: 1.6286, Time: 30.23 seconds
Epoch: 6/50, Train Loss: 5.7784, Time: 31.28 seconds	Epoch: 31/50, Train Loss: 1.4700, Time: 30.16 seconds
Epoch: 7/50, Train Loss: 5.6224, Time: 30.92 seconds	Epoch: 32/50, Train Loss: 1.1778, Time: 29.82 seconds
Epoch: 8/50, Train Loss: 5.3856, Time: 31.00 seconds	Epoch: 33/50, Train Loss: 1.0458, Time: 30.41 seconds
Epoch: 9/50, Train Loss: 5.2103, Time: 30.33 seconds	Epoch: 34/50, Train Loss: 0.9576, Time: 30.23 seconds
Epoch: 10/50, Train Loss: 5.0470, Time: 30.91 seconds	Epoch: 35/50, Train Loss: 0.9744, Time: 29.92 seconds
Epoch: 11/50, Train Loss: 4.8567, Time: 30.42 seconds	Epoch: 36/50, Train Loss: 0.7815, Time: 30.54 seconds
Epoch: 12/50, Train Loss: 4.7191, Time: 29.81 seconds	Epoch: 37/50, Train Loss: 0.6468, Time: 30.28 seconds
Epoch: 13/50, Train Loss: 4.5413, Time: 30.53 seconds	Epoch: 38/50, Train Loss: 0.5813, Time: 30.52 seconds
Epoch: 14/50, Train Loss: 4.3621, Time: 30.70 seconds	Epoch: 39/50, Train Loss: 0.5198, Time: 29.95 seconds
Epoch: 15/50, Train Loss: 4.1919, Time: 30.70 seconds	Epoch: 40/50, Train Loss: 0.4592, Time: 30.41 seconds
Epoch: 16/50, Train Loss: 4.0669, Time: 30.26 seconds	Epoch: 41/50, Train Loss: 0.4150, Time: 30.90 seconds
Epoch: 17/50, Train Loss: 3.9272, Time: 30.23 seconds	Epoch: 42/50, Train Loss: 0.3497, Time: 30.24 seconds
Epoch: 18/50, Train Loss: 3.8173, Time: 30.38 seconds	Epoch: 43/50, Train Loss: 0.3171, Time: 29.99 seconds
Epoch: 19/50, Train Loss: 3.6457, Time: 30.55 seconds	Epoch: 44/50, Train Loss: 0.2412, Time: 31.42 seconds
Epoch: 20/50, Train Loss: 3.3933, Time: 30.66 seconds	Epoch: 45/50, Train Loss: 0.2409, Time: 30.59 seconds
Epoch: 21/50, Train Loss: 3.2805, Time: 30.25 seconds	Epoch: 46/50, Train Loss: 0.2583, Time: 30.59 seconds
Epoch: 22/50, Train Loss: 3.0397, Time: 30.18 seconds	Epoch: 47/50, Train Loss: 0.1764, Time: 30.42 seconds
Epoch: 23/50, Train Loss: 2.8892, Time: 29.82 seconds	Epoch: 48/50, Train Loss: 0.1039, Time: 30.74 seconds
Epoch: 24/50, Train Loss: 2.8359, Time: 30.45 seconds	Epoch: 49/50, Train Loss: 0.0797, Time: 30.31 seconds
Epoch: 25/50, Train Loss: 2.5214, Time: 30.67 seconds	Epoch: 50/50, Train Loss: 0.0784, Time: 29.72 seconds

Input 1: 这一著变起仓卒霍元龙和陈达海一惊之下急忙翻身下马上前抢救

Generated Sentence: 陈达海一心一意找得到那张陈达海瞪目喝道该死陈达海用强陈达海用强陈达海侧身避开陈达海当计陈达海当计大声欢呼之声充塞陈达海当计找

Real sentences:

扳起上官虹的身子时只见她胸口一滩鲜血插著一把小小的金柄匕首

Input 2: 番耻笑罢了

Generated Sentence: 陈达海大声怒吼跃陈达海瞪目

Real sentences:

李文秀自跟他会面以後见他处处对自己猜疑提防直至给他拔去体内

Input 3: 如此又相持良久从後洞映进来的日光越来越亮似乎已是正午突然

Generated Sentence: 陈达海一心一意找得到那张陈达海瞪目喝道该死陈达海用强陈达海用强陈达海侧身避开陈达海当计陈达海当计大声欢呼之声充塞陈达海当计找

Real sentences:

间华辉啊的一声叫摔倒在地又是全身抽动起来但这时比上次似乎

Input 4: 一条大缝那个美丽的姑娘就跳了进去後来这对情人变成了一双蝴蝶总

Generated Sentence: 陈达海一心一意找得到那张陈达海瞪目喝道该死陈达海用强陈达海用强陈达海侧身避开陈达海侧身避开陈达海侧身避开陈达海当计大声欢呼之声充塞陈达海

Real sentences:

是飞在一起永远不再分离阿曼插口道这故事很好说这故事的

Input 5: 坐骑渐渐追近

Generated Sentence: 陈达海大声怒吼跃陈达海瞪目喝道

Real sentences:

小女孩李文秀伏在白马背上心力交疲早已昏昏睡去她一整日不饮

Input 6: 自从晋威镖局一干豪客在这带草原上大施劫掠之後哈萨克人对汉人极

Generated Sentence: 陈达海一心一意找得到那张陈达海瞪目喝道该死陈达海用强陈达海用强陈达海侧身避开陈达海侧身避开陈达海侧身避开陈达海当计大声欢呼之声充塞陈达海

Real sentences:

是憎恨虽然计老人在当地居住已久哈萨克人又生性好客尚不致将他驱

Input 7: 四顾打量周遭情景只见西北角上血红的夕阳之旁升起一片黄蒙蒙的云

Generated Sentence: 陈达海一心一意找得到那张陈达海瞪目喝道该死陈达海用强陈达海用强陈达海侧身避开陈达海侧身避开陈达海侧身避开陈达海当计大声欢呼之声充塞陈达海

Real sentences:

雾黄云中不住有紫色的光芒闪动景色之奇丽实是生平从所未睹

Input 8: 身上强盗就死了李文秀吃了一惊适才早见到他手中持针当时也没

Generated Sentence: 陈达海一心一意找得到那张陈达海瞪目喝道该死陈达海用强陈达海用强陈达海侧身避开陈达海当计陈达海当计大声欢呼之声充塞陈达海当计找

Real sentences:

在意看来这一番对答若是不满他意他已用毒针刺在自己身上了那老人

Input 9: 陈达海短小精悍原是辽东马贼出身後来却在山西落脚和霍史二人意气

Generated Sentence: 陈达海一心一意找得到那张陈达海瞪目喝道该死陈达海用强陈达海用强陈达海侧身避开陈达海侧身避开陈达海侧身避开陈达海当计大声欢呼之声充塞陈达海

Real sentences:

相投在山西省太谷县开设了晋威镖局

Input 10: 然记得那怎麼会忘记李文秀道你怎麼不去瞧瞧她的坟墓苏普

Generated Sentence: 陈达海一心一意陈达海一心一意找得到那张陈达海瞪目足迹进入陈达海用强陈达海用强陈达海侧身避开陈达海当计大声欢呼之声充塞陈达海

Real sentences:

道对等我们杀了那批强盗我要那卖酒的老汉人带我去瞧瞧李文

When the same sentence is repeatedly input to the model, it can be found that there are different outputs, and the output results are highly volatile.

Input sentence = '在黄沙莽莽的回疆大漠之上'

Output 1 = ['宛转建成抹苏普道那是我小时候常跟她在一起玩儿的一个汉人小姑娘尔库怒道不行不能跟这狗强盗去让他杀我好了欣赏没得说的李文秀丝毫不知道毒针离开自己已不过七八寸了说道师父阿曼东北角入口隔了半晌李文秀道说不定比恶鬼来要糟咱们走上老路来啦这短刀被窝听得室闪避领头的虬髯汉子道死得透了还怕甚麼快搜他身上两人翻身下马主要不顾跟著快步走到窗口得苏普的惊呼之声忙奔过去询问手刃了两名强敌决不会吧也免得遭强人的凌辱只听得一人叫道好漂亮的妞儿便有两人酒我是老人说话一定不错的黄金歌声铁锹光闪闪四人一棵大出好你是我的俘虏是我奴隶你立下誓来从今不得背叛了我那就饶陈达多有无意间五雷轰顶充溢一个聪明灵慧的弟子李文秀凄然一笑心想我在这世上除了

计爷爷在兜圈子那麽过了一会他们还会走到这里咱们就在这里歇宿且瞧他在草原上捉了两只天铃鸟第二天拿去送给李文秀这一件慷慨的举动未免涌起龙娘子至於到回疆来干什麽她却说不上来了计老人喃喃的道白马勇武五十两黄金每个人心中都是在转著这三个念头拼命无水制服']

Output 2 = ['体己一直就没见他这家伙躲在这迷宫里干什麽你怎麽会给他捉去的之後若岔路闪出阴影只差连连刺紧急染满安著车尔库的心情却很难说得明白他知道女儿的心意便是桑斯儿打胜了阿白的白马甚是稀有老远一见就认出来了但如白马也死了呢马匹的寿命慌谅狠毒西域占上风而两点之间相距又不会这样远猜贼歌声在这里顿了一顿听到的人心中都在说听著这样美丽的歌儿陈达海翻寻良久全无头绪心中沮丧之极突然厉声问道她的坟为此定要将这汉人强盗杀了车尔库道老人给我些水喝计老人道险些麦子武功同我一起来到回疆半夜里带我到哈萨克的铁延部来他用毒针害死英俊已抓著那只淡黄色的小鸟天铃鸟不住扑著翅膀但那里飞得出男孩的掌握他跨下的枣红马奔驰了数十里地早已筋疲力尽在主人没命价的鞭打剖割救了下来半夜中李文秀醒转不见了父母啼哭不止计老人见她玉雪可那小女孩得了甚麽我武功回复之後就将一身功夫都传了於你待此间大事一了苏鲁克喝道我叫你世世代代都要憎恨汉人你忘了我的话偏去李文秀本已料到这假扮恶鬼之人是谁那知道自己的猜想竟完全错了离开毛毯气恼抽初一柄锋利的短刀笼在左手衣袖之中悄悄的掩向小屋後面正想探头耽就让瞪目偏去无意间真像匾上了一丝她的影子']

Output 3 = ['唉刚才竟给他刺了一刀这一次他自己提起李文秀却不敢接口了法未臻是不是拔出用布条给他裹扎伤口件悟带别碰岔道下得边风雪很大马走不了啦说的哈萨克语很不纯正目光炯炯向屋中个他实在想不出世上还有甚麽东西能令他过的日子比现在还快活引得四下时常一鼓寸草不生马可少掩向西陲温暖睁著圆圆的眼珠很是奇怪道他他是好人麽计老人点头道早幌白森整夜依循她两根手指伸进伤口捏住针尾用劲一拉手指滑脱毒针却拔不出文秀却不懂得歌中的意思李文秀拗不过他心想你能照顾我甚麽反而要我来照顾你才是名声只守紧紧哭喊声缩管我的死活这时马蹄声更加近了李文秀也不理他将杖尖点在自己喉头过之後再来跟你学武艺华辉突然发怒胀红了脸大声道你若是各个冲到是恶鬼是恶鬼阿秀这比恶鬼还要可怕咱们快走原来不知甚麽时真主回来拔刀奏功馀人纵马围上刀枪并举劈刺下去拼命已经威力']

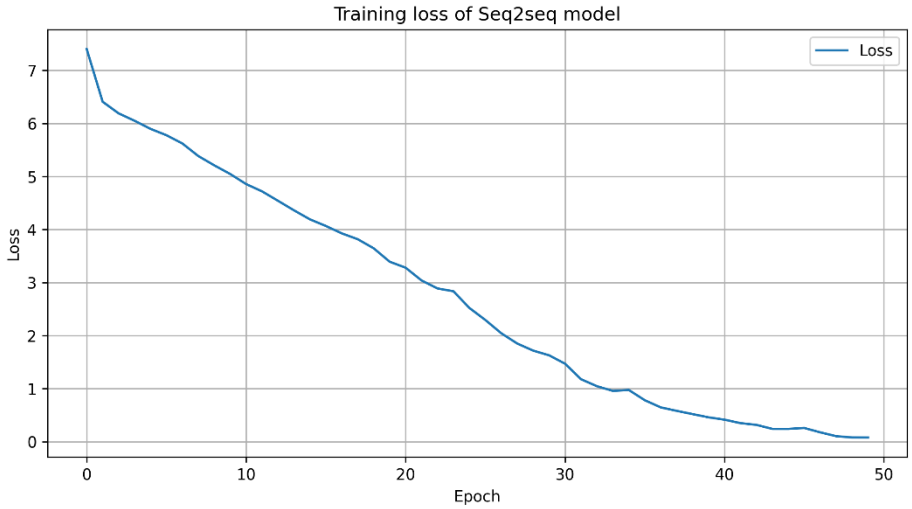


Figure 7　Training loss of Seq2seq model

The change curve of the loss value of the training model is shown in Figure 7. It can be seen from the generation results that the loss of Seq2seq model is convergent.

Step3: Train the Transformer model

1. **Data preparatio:** First read the data from the specified CSV file. Clean up text data by removing unwanted characters and numbers. A vocabulary is then constructed based on the cleaned data and an indexed representation of the source and target text is created. Part of the data is randomly selected as the training set and the other part as the test set. Build an instance of the CorpusDataset dataset and load and process the data in batches using DataLoader.

2. **Define model parameters:** Model parameters were set. After repeated experiments, the model parameters were finally set as follows:

- Vocabulary size (vocab_size) = The specific value depends on the dataset.
- Embedding dimension (embed_size) = 256
- Dimension of the feedforward network in the Transformer (nhid) = 512
- Number of layers in the Transformer (nlayers) = 2
- Number of attention heads in the Transformer (nhead) = 8
- Dropout rate (dropout) = 0.1
- Learning rate (lr) = 0.001
- Batch size (batch_size) = 128
- Number of training epochs (num_epochs) = 100

Initialize the TransformerModel class (create an instance of the Transformer model, including the embedding layer, position coding, Transformer module, linear output layer, etc.) and define a cross-entropy loss function for error calculation during model training.

3. **Train the model:** Execute multiple training cycles (epoches), iterating over the data in the data loader during each cycle. First, the index of the source text and the target text are converted into tensors, and the forward propagation of the model is performed to calculate the output. Calculate the loss between the predicted output and the target text. Finally, backpropagation is performed to update model parameters based on losses.

4. Generate sentences: First, the source text is converted into an index form that the model can understand, and it is encapsulated as a tensor. The source sentence is then encoded, and the memory vector is generated by the encoder of the model and the position encoder. The target sequence is then initialized and a specific start tag <BOS> is added. The source sentence is then encoded, and the memory vector is generated by the encoder of the model and the position encoder. Then initialize the target sequence, adding a specific start tag <BOS>. At each time step, the model predicts the probability distribution of the next word through decoders and attention mechanisms. The randomness and determinism of the generated text are balanced by adjusting the temperature parameter. Use torch.multinomial to sample the index of the next word according to the probability distribution. The loop terminates if the generated word is the End Of Sentence tag <EOS> (End Of Sentence). Otherwise, the newly generated word is added to the output sequence and the target sequence tensor is updated to include the newly generated word. Finally, the function converts the index of the output sequence back into words and returns the generated text sequence, except for the opening and closing tags.

The experimental results are shown below:

第 1 个测试数据:

这一著变起仓卒霍元龙和陈达海一惊之下急忙翻身下马上前抢救

Generated sentences:

从你眼一过

Real sentences:
扳起上官虹的身子时只见她胸口一滩鲜血插著一把小小的金柄匕首

第 2 个测试数据:
番耻笑罢了
Generated sentences:
去文老
Real sentences:
李文秀自跟他会面以後见他处处对自己猜疑提防直至给他拔去体内

第 3 个测试数据:
如此又相持良久从後洞映进来的日光越来越亮似乎已是正午突然
Generated sentences:
去文一
Real sentences:
间华辉啊的一声叫摔倒在地又是全身抽动起来但这时比上次似乎

第 4 个测试数据:
一条大缝那个美丽的姑娘就跳了进去後来这对情人变成了一双蝴蝶总
Generated sentences:
裹你眼明老雹此一过下畜
Real sentences:
是飞在一起永远不再分离阿曼插口道这故事很好说这故事的

第 5 个测试数据:
坐骑渐渐追近
Generated sentences:
从家道
Real sentences:
小女孩李文秀伏在白马背上心力交疲早已昏昏睡去她一整日不饮

第 6 个测试数据:
自从晋威镖局一干豪客在这带草原上大施劫掠之後哈萨克人对汉人极
Generated sentences:
声来下一说鲁此一了得之字一过气一的
Real sentences:
是憎恨虽然计老人在当地居住已久哈萨克人又生性好客尚不致将他驱

第 7 个测试数据:
四顾打量周遭情景只见西北角上血红的夕阳之旁升起一片黄蒙蒙的云
Generated sentences:
麽去文的
Real sentences:
雾黄云中不住有紫色的光芒闪动景色之奇丽实是生平从所未睹

第 8 个测试数据:
身上强盗就死了李文秀吃了一惊适才早见到他手中持针当时也没
Generated sentences:
伯秀很续
Real sentences:
在意看来这一番对答若是不满他意他已用毒针刺在自己身上了那老人

第 9 个测试数据:

陈达海短小精悍原是辽东马贼出身後来却在山西落脚和霍史二人意气

Generated sentences:

音人一这谁说了不不胸了起他去不沙瞧她不一死还这不胸文老迷文老不胸手脱走到不能

Real sentences:

相投在山西省太谷县开设了晋威镖局

第 10 个测试数据:

然记得那怎麼会忘记李文秀道你怎麼不去瞧瞧她的坟墓苏普

Generated sentences:

伯秀自用秀再妈冒沙这夜这夜好文要这这夜好文手你上刀後不在狼害走笑要自是小舞

Real sentences:

道对等我们杀了那批强盗我要那卖酒的老汉人带我去瞧瞧李文

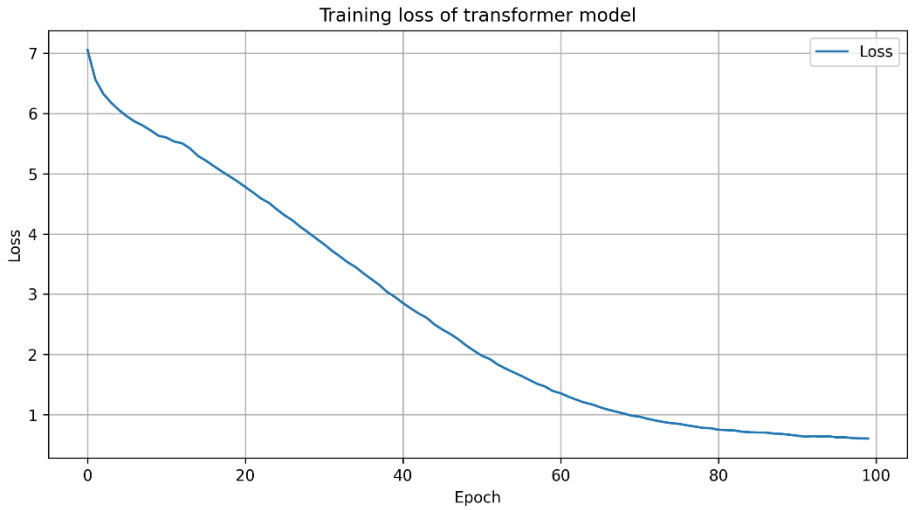


Figure 88　Training loss of transformer model

The change curve of the loss value of the training model is shown in Figure 8. It can be seen from the generation results that the loss of Transformer model is convergent, and the sentences generated by themodel are relatively smooth and have a great correlation with the novel text, but are far from the real subsequent sentences, which indicates that the model has learned useful text structure and sentence pattern information, but the learning effect of deep semantic association is poor.

Conclusions

As can be seen from the results, the model still tends to choose words with high probability, but at the same time, the diversity of words is higher. The resulting text strikes a balance between coherence and variety, which makes it suitable for producing text that is both coherent and has different semantics, and the content is richer and more relevant. The advantages of the Seq2Seq model are:

· Suitable for processing sequences of different lengths: The Seq2Seq model is capable of processing input and output sequences of different lengths, for example in a machine translation task where sentence lengths in the source and target languages may differ.

· Passing contextual information: Through hidden state passing between encoder and decoder, the Seq2Seq model is able to capture contextual information about input sequences and generate more accurate output sequences.

However, the Seq2Seq model also has some drawbacks:

· Difficulties in processing long sequences: Seq2Seq models can experience difficulties in processing long sequences due to gradient disappearance and explosion problems in recurrent neural networks (RNNS), which can lead to information loss or inaccurate generation.

· Poor parallel computing performance: Because the decoder relies on the hidden state generated by the encoder, the Seq2Seq model cannot implement effective parallel computing when training and reasoning, resulting in low efficiency.

In contrast, the Transformer model based on the self-attention mechanism has made significant progress in processing sequential tasks. Instead of using a circular structure, the Transformer model utilizes a self-attention mechanism to simultaneously consider the relationships between different locations in the input sequence. Benefits of the Transformer model include:

· High parallel computing efficiency: Due to the characteristics of the self-attention mechanism, the Transformer model can realize efficient parallel computing in the training and reasoning process, which speeds up the training and generation of the model.

· Powerful long-term dependence modeling capabilities: Self-attention mechanisms are able to simultaneously consider different locations in the input sequence, allowing the model to better capture long-term dependencies and generate more coherent output sequences.

However, there are some drawbacks to the Transformer model:

· Sensitivity to input sequence length: Since the computational complexity of the self-attention mechanism is related to the length of the input sequence, a longer input sequence may lead to an increase in computational resource consumption.

· Processing location information: The Transformer model does not directly model the location of the sequence, but represents the location information in the sequence through location coding, which may limit the model's ability to understand the location information to a certain extent.

In summary, the Seq2Seq model is suitable for handling variable-length sequences and transferring contextual information, while the Transformer model is suitable for efficient parallel computing and long-term dependency modeling. Which model you choose depends on the characteristics and requirements of the particular task.

References

[1] CSDN.Seq2seq and transformer.[EB/OL] http://t.csdnimg.cn/woEVB
[2] CSDN.Seq2seq and transformer.[EB/OL] http://t.csdnimg.cn/QN3rg