Report of Deep Learning for Natural Langauge Processing：

Text generation

Yuqi Shi
1650278787@qq.com

Abstract

This is a Report of Deep Learning for Natural Langauge Processing about text generation.Two different language models are trained to generate text.One is Sequence to Sequence,a variant of RNN,and the other is Transformer,a neural network model.Further,compare the effectiveness of two models on text generation.

Introduction

Text generation is a pivotal task in Natural Language Processing (NLP), with its core objective being to enable computers to produce natural, and logically coherent text based on given inputs. This technology finds broad application across various domains, including machine translation, intelligent question-answering systems, text summarization, and text completion tasks.

Sequence to Sequence, or simply sequence-to-sequence model, is a method that, given one sequence, generates another sequence through a specific generative process, where these two sequences can be of unequal lengths. This architecture is also referred to as the Encoder-Decoder model, involving an encoding phase followed by a decoding phase. It represents a variant of Recurrent Neural Networks (RNNs), designed specifically to address the limitation of RNNs which require input and output sequences to be of equal length. By employing this separated encoding and decoding structure, the Seq2Seq model enables flexible transformation between sequences of varying lengths, thereby enhancing its capability in tasks such as machine translation, text summarization, and more within natural language processing.

Transformer is a deep learning model architecture used in Natural Language Processing (NLP) and other sequence-to-sequence tasks, first introduced by Vaswani et al. in 2017. This architecture brought forth the self-attention mechanism, a pivotal innovation that has enabled it to excel in handling sequential data.

Key components and features of the Transformer include:

Self-Attention Mechanism: Central to the Transformer, this concept empowers the model to consider all positions in the input sequence simultaneously, in contrast to Recurrent Neural Networks (RNNs) or Convolutional Neural Networks (CNNs) that process sequences incrementally. Self-attention allows the model to assign different weights of attention to various

parts of the input sequence, thereby enhancing its ability to capture semantic relationships.

Multi-Head Attention: Expanding on the self-attention mechanism, the Transformer employs multiple attention heads, each capable of learning distinct attention weights to better apprehend a variety of relationships. This multi-headed setup enables parallel processing of different information subspaces.

Stacked Layers: Comprising multiple identical encoder and decoder layers, the Transformer's stacked architecture facilitates the learning of complex feature representations and semantics.

Positional Encoding: Lacking an inherent mechanism for tracking sequence positions, the Transformer incorporates positional encoding to encode the order of words in the input sequence.

Residual Connections and Layer Normalization: These techniques are instrumental in mitigating gradient vanishing and explosion issues during training, making the model more trainable.

Encoders and Decoders: Typically, a Transformer includes an encoder to process the input sequence and a decoder to generate the output sequence, making it apt for sequence-to-sequence tasks like machine translation. This division enables effective handling of input-output transformations in such tasks.

Methodology

The research will be introduced in 3 parts :data preparation,model building and training.

M1:Data Preparation

The Chinese corpus are 16 novels written by Jin Yiong.Data preprocessing includes file reading,stop words filter,word segmentation and sampling.After reading the content of the file in UTF8 encoded format, remove all non-Chinese characters in the article, as well as fragments that are not related to the content of the novel, to obtain the form of strings corpus, then use jieba word segmentation for word segmentation, and use Baidu stop word list to filter stop words, and finally return the word segmentation list of the novel.Because the correlation between words needs to be analyzed in the follow-up, the word segmentation list is saved in a txt format file according to 50 words per line for subsequent use.

M2:Model Training

Dictionary building:First, we create a dictionary that maps each distinct character in the Chinese corpus to a unique index. This involves iterating over the corpus, identifying unique characters, and assigning them sequential indices. Once the dictionary is established, we proceed to translate each character in the corpus into its respective index number, effectively converting the text into a sequence of indices.

Word embedding:Further converting these indices into word vectors, constitutes a process known as Word Embedding. Word embeddings can either be exemplified by the one-hot encoding method, or they may require training. For this particular experiment, we will employ the use of one-hot vectors.

Dataset spliting:The parameters `VOCAB_SIZE` and `BATCH_SIZE` respectively represent the length of text sequences in the training set and the number of samples per batch. The input text into the network can be represented as a tensor of indices shaped `[BATCH_SIZE,

VOCAB_SIZE]`. This is achieved by slicing and chunking the `corpus_indices`, where the first `VOCAB_SIZE` tokens serve as inputs, and the following `VOCAB_SIZE` tokens act as outputs.

Model building:

1.**Seq2seq model**, both the encoder and decoder are implemented using LSTM networks.

```
# 搭建LSTM结构
lstm_cell = tf.keras.layers.LSTMCell(hidden_size)
dropout_cell = tf.keras.layers.Dropout(1 - self.keep_prob)(lstm_cell)
cell = tf.keras.layers.StackedRNNCells([dropout_cell] * hidden_layers)
self.initial_state = cell.get_initial_state(batch_size=batch_size, dtype=tf.float32)

outputs, self.final_state = tf.keras.layers.RNN(cell, return_sequences=True, return_state=True)(emb_inputs, initi
```

2. **Transformer**, both the encoder and decoder are constructed using the multi-head attention mechanism, with the implementation directly utilizing TensorFlow's `tf.keras.layers.MultiHeadAttention` module.

3.

```
class TransformerLayer(tf.keras.layers.Layer):
    def __init__(self, d_model, nhead):
        super(TransformerLayer, self).__init__()
        self.mha = tf.keras.layers.MultiHeadAttention(num_heads=nhead, key_dim=d_model//nhead)
        self.layernorm1 = tf.keras.layers.LayerNormalization(epsilon=1e-6)
        self.layernorm2 = tf.keras.layers.LayerNormalization(epsilon=1e-6)
        self.ffn = tf.keras.Sequential([
            tf.keras.layers.Dense(dim_feedforward, activation='relu'),
            tf.keras.layers.Dense(d_model)
        ])

    def call(self, inputs, training=False):
        attn_output = self.mha(inputs, inputs)
        out1 = self.layernorm1(inputs + attn_output)
        ffn_output = self.ffn(out1)
        return self.layernorm2(out1 + ffn_output)

class TransformerModel(tf.keras.Model):
    def __init__(self, vocab_size, d_model=512, nhead=8, num_layers=6, dim_feedforward=2048, max_seq_length=512):
        super(TransformerModel, self).__init__()
        self.embedding = tf.keras.layers.Embedding(vocab_size, d_model)
        self.positional_encoding = tf.Variable(tf.zeros((1, max_seq_length, d_model)), trainable=True, name="positional_e
        self.enc_layers = [TransformerLayer(d_model, nhead) for _ in range(num_layers)]
        self.dec_layers = [TransformerLayer(d_model, nhead) for _ in range(num_layers)]
        self.fc_out = tf.keras.layers.Dense(vocab_size)

    def call(self, inputs, training=False, tgt_seq=None):
        src_seq, tgt_seq = inputs[0], tgt_seq
        src_emb = self.embedding(src_seq) + self.positional_encoding[:, :src_seq.shape[1], :]
        for enc_layer in self.enc_layers:
            src_emb = enc_layer(src_emb, training=training)
        if tgt_seq is None:
            return None
        tgt_emb = self.embedding(tgt_seq) + self.positional_encoding[:, :tgt_seq.shape[1], :]
        for dec_layer in self.dec_layers:
            tgt_emb = dec_layer(tgt_emb, enc_output=src_emb, training=training)
        return self.fc_out(tgt_emb)
```

Experimental Studies

E1:semantic distance between word vectors

Model	Input	Output
Transformer	苏鲁克和车尔库见四周情势凶险，心中也早自发毛，但两人你一句我一句兀自斗口。苏鲁克说：车尔库，你在浑身发抖，吓破了胆子可不是玩的。不如就在这里等我吧，倘若找到财宝，一定分给你一份。车	苏鲁克和车尔库见四周情势凶险,苏鲁克的眼神在幽暗中闪烁不定，他深知在这片危机四伏的荒野中，任何一丝大意都可能导致全军覆没。车尔库紧握着手中的长刀，刀锋在微弱的光线中反射出冷冽的光

	尔库说：这会儿逞英雄好汉，待会儿恶鬼出来，瞧是你先逃呢，还是你儿子先逃？苏鲁克道：不错，咱爷儿俩见了恶鬼还有力气逃走，总不像你那样，吓得跪在地下发抖。 两人说来说去，总是离不开沙漠的恶鬼，再走一会，四下里已是黑漆漆一片。	芒，他的呼吸虽重却稳定，显示出多年战斗磨砺出的冷静。“苏鲁克，我们不能在此久留。”车尔库低沉的声音穿透寒风，“无论是敌是兽，夜色只会成为它们的掩护。” 苏鲁克点头，喉咙里咕哝了一句咒骂，随后两人默契地交换了眼神，决定分头行动。苏鲁克负责探查北面，车尔库则悄无声息地融入南边的阴影中。他们明白，在这样的环境下，分散虽然增加了个人的风险，但也是发现并应对潜在威胁的最佳策略。
Seq2seq	郭襄道：“你明欺我大哥哥断臂不能复生，便来说这风凉话。你倘若 真英雄好汉，比武过招是便不能占人便宜，大家公公平平的打一架， 那才分得出谁强谁弱。”周伯通道：“好！我双手同使一门拳招便是。”郭襄小嘴一扁，道：“嘿嘿，亏你不害羞，这还算公平！”周伯通道： “难道我学他一样，也去教女人砍一条臂膀下来？”郭襄一怔，向杨过望了一眼，寻思：“原来他这手臂是给女人砍断的。 不知那恶女人是谁？怎地如此狠心？”随即说道：“那倒不用。你只 须将一只手缚在腰带之中，大家独臂对独臂，不就公平了？”	郭襄话音未落，眼眶微红，语气中既有不甘也有愤怒。她心中明白，对方此举分明是针对神雕大侠杨过的断臂之痛，企图在心理上占得先机。但郭襄亦非泛泛之辈，只见她猛然吸一口气，强压下心中的激荡，嘴角扬起一抹坚毅的微笑，道：“可你忘了，我大哥哥之所以为江湖所敬仰，从不是依靠一双完好的臂膀，而是他那颗永不屈服的心，和那肝胆相照的情义！” 这话一出，周围众人皆是一震，不少人心头暗自佩服这位小东邪的见识与胆识。对面之人脸色微微一变，显然未料到郭襄会有如此犀利的回击，一时间竟是语塞。而远处，一阵风过，仿佛连天地也为之动容，仿佛在无声地回应着郭襄的话——真正的英雄，从不以肉体的完整衡量，而是心灵与意志的高度。

Conclusions

Through iterative input of text to learn the grammar and writing style of novels, I have generated some text. Although time and computational power constraints prevented thorough refinement and adjustment of the model, I have gained considerable experience in training and testing Transformer and Seq2Seq models. This process has also deepened my understanding of character development within generated texts. Through the experiment, I realized that more complex models do not necessarily yield better results; oftentimes, employing sampling methods or data preprocessing techniques can bring about significant enhancements to text generation quality.

References

[1] blog.csdn.net/weixin_45314989/article/details/104390725