Week5

19377419 孙启航

作业要求

> 实现文本处理的Tokenizer类
> – 深度学习处理自然语言时，会常常用到Tokenizer（https://huggingface.co/transformers/tokenizer_summary.html）。简单来说，就是按照预先定义好的词典，把文本编码成整数序列的过程。深度学习模型进行文本挖掘任务时会经常需要处理这种编码过的序列。在构建过程中，有时候我们希望句子的长度能够整齐，所以会规定一个特殊的号码[PAD]=0，代表填充位。具体地，例如词典为{'[PAD]': 0, '我':1, '是': 2, '北京': 3, '大学生': 4, '的': 5 }，那么"我是北京的大学生" 将被编码为[1, 2, 3, 5, 4]；如果需要句子长度为8，那么我们在后面填充[PAD]，得到[1,2,3,5,4,0,0,0]。现在请编程实现一个基础的中文Tokenizer类，分别实现按字（深度学习中也常用该方式）或按词（通过jieba分词）进行编码，并在前次作业提供的评论或微博文本数据集上进行测试。
> – 1. **init**(self, chars,coding='c',PAD=0) 输入将要需要操作的文本（一个字符串的列表），这里需要完成词典的构建（即汉字到正整数的唯一映射的确定）。注意构建词典一是要根据coding来选择按词构建（coding='w'），还是按字构建，默认按字构建；PAD默认为0。
> – 2. tokenize(self, sentence) 输入一句话，返回分词(字）后的字符列表(list_of_chars)。
> – 3. encode(self, list_of_chars) 输入字符(字或者词）的字符列表，返回转换后的数字列表(tokens)。
> – 4. trim(self, tokens, seq_len) 输入数字列表tokens，整理数字列表的长度。不足seq_len的部分用PAD补足，超过的部分截断。
> – 5. decode(self, tokens) 将模型输出的数字列表翻译回句子。如果有PAD，输出'[PAD]'。
> – 6. encode(self，seq_len) 返回所有文本（chars)的长度为seq_len的tokens。
> – 7. 附加：seq_len是句子的长度，实际任务中一般怎么来确定一个合适的长度，请以前次作业中的评论文本或微博文本为例，通过文本的长度分布来进行观察和讨论。
> – 8. 附加：了解一下基于Tokenizer编码后的序列在序列模型（如RNN, LSTM, GRU,Transformer)中一般是如何使用的，可用来完成哪些任务。

代码实现

```
import jieba

class Tokenizerpro:
    '''
    将文本编码成整数序列
    '''

    def __init__(self, chars, coding='c', PAD=0):
        '''默认情况下是按照字符构建字典'''
        self.chars = chars
        if coding == 'c':
            self.coding = coding
            self.dic = {}
            self.dic['[PAD]'] = 0
            flag = 1

            for i in chars:
                for j in i:
                    if j not in self.dic.keys():
```

```
                    self.dic[j] = flag
                    flag += 1
                else:
                    continue

        elif coding == 'w':
            self.coding = coding
            self.dic = {}
            self.dic['[PAD]'] = 0
            flag = 1
            for i in chars:
                lis = jieba.lcut(i)
                for j in lis:
                    if j not in self.dic.keys():
                        self.dic[j] = flag
                        flag += 1
                else:
                    continue

        else:
            pass

    def tokenize(self, sentence):
        '''
        根据字典的创建规则，对文本进行处理

        如果coding为c，那么就不进行处理
        如果coding为w，那么利用jieba库进行分词处理'''
        list_of_chars = []
        if self.coding == 'c':
            pass
        elif self.coding == 'w':
            sentence = jieba.lcut(sentence)
        for i in sentence:
            list_of_chars.append(i)

        return list_of_chars

    def encode(self, list_of_chars):
        '''
        利用生成的字典，对文本进行编码处理
        '''
        tokens = []
        for i in list_of_chars:
            tokens.append(self.dic[i])

        return tokens

    def trim(self, tokens, seq_len):
        '''
        整理数字列表的长度，不足的用0补齐，超过的部分则进行切片处理
        '''
        if len(tokens) > seq_len:
            return tokens[:seq_len]
        else:
            tokens.extend([0] * (seq_len - len(tokens)))
            return tokens
```

```
    def decode(self, tokens):
        '''
        将处理后的数字列表翻译成文本，其中0对应的字符为[PAD]
        '''
        dic = dict(zip(self.dic.values(), self.dic.keys()))
        sentence = ''
        for i in tokens:
            if i != 0:
                sentence += dic[i]
            else:
                sentence += '[PAD]'
        return sentence

    def encode_len(self, seq_len):
        '''
        返回整个文档中所有文本的长度为seq_len的数字序列
        '''
        seq_len_list = []
        for i in self.chars:
            seq_len_list.append(
                self.trim(self.encode(self.tokenize(i)), seq_len))
        return seq_len_list


def main():
    file = 'BUAA_21/Week5/jd_comments.txt'

    with open(file, 'r', encoding='utf-8') as f:
        lines = f.readlines()

    with open("BUAA_21/Week5/output.txt", "w", encoding='utf-8') as output:

        lines = [i.strip('\n') for i in lines]

        new = Tokenizerpro(lines)

        # 以京东文档中第二个评论为例
        list_of_chars = new.tokenize(lines[1])
        print("List of chars (The second comment):\n",
              list_of_chars, "\n\n", file=output)

        tokens = new.encode(list_of_chars)
        print("Token of the second comment):\n", tokens, "\n\n", file=output)

        new_tokens = new.trim(tokens, 150)
        print("After trim function:\n", new_tokens, "\n\n", file=output)

        sentence = new.decode(new_tokens)
        print("Sentence:\n", sentence, "\n\n", file=output)

        seq_len_list = new.encode_len(100)
        print("Seq_len_list:", *seq_len_list, "\n\n", sep='\n', file=output)

if __name__ == '__main__':
    main()
```

输出结果示例

```
List of chars (The second comment):
 ['性', '能', '挺', '不', '错', '的', '，', '这', '么', '小', '个', '头', '这', '个',
'性', '能', '蛮', '不', '错', '，', '装', '上', '后', '电', '脑', '复', '活', '了',
'，', '原', '先', '卡', '死', '了', '，', '现', '在', '开', '机', '7', '秒', '左',
'右', '非', '常', '快', '，', '这', '么', '大', '容', '量', '装', '游', '戏', '什',
'么', '的', '美', '滋', '滋', '。', 'M', 'L', 'C', '颗', '粒', '寿', '命', '有',
'保', '证', '，', '看', '到', 'T', 'L', 'C', '就', '讨', '厌', '，', '听', '说',
'以', '后', '主', '流', 'T', 'L', 'C', '，', '主', '打', '性', '价', '比', '的',
'会', '是', 'Q', 'L', 'C', '寿', '命', '更', '低', '，', '所', '以', 'M', 'L', 'C',
'颗', '粒', '以', '后', '难', '买', '了', '，', '价', '格', '不', '知', '道', '得',
'多', '贵', '啊']


Token of the second comment):
 [45, 46, 47, 22, 23, 8, 9, 48, 35, 28, 49, 50, 48, 49, 45, 46, 51, 22, 23, 9,
52, 53, 54, 55, 56, 57, 58, 59, 9, 60, 61, 62, 63, 59, 9, 33, 64, 18, 12, 65,
66, 67, 68, 69, 70, 71, 9, 48, 35, 72, 73, 74, 52, 75, 76, 34, 35, 8, 77, 78,
78, 44, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 9, 89, 90, 91, 80, 81, 92, 93,
94, 9, 95, 96, 41, 54, 11, 97, 91, 80, 81, 9, 11, 98, 45, 99, 100, 8, 101, 102,
103, 80, 81, 84, 85, 104, 105, 9, 106, 41, 79, 80, 81, 82, 83, 41, 54, 107, 108,
59, 9, 99, 109, 22, 110, 111, 112, 113, 114, 115]


After trim function:
 [45, 46, 47, 22, 23, 8, 9, 48, 35, 28, 49, 50, 48, 49, 45, 46, 51, 22, 23, 9,
52, 53, 54, 55, 56, 57, 58, 59, 9, 60, 61, 62, 63, 59, 9, 33, 64, 18, 12, 65,
66, 67, 68, 69, 70, 71, 9, 48, 35, 72, 73, 74, 52, 75, 76, 34, 35, 8, 77, 78,
78, 44, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 9, 89, 90, 91, 80, 81, 92, 93,
94, 9, 95, 96, 41, 54, 11, 97, 91, 80, 81, 9, 11, 98, 45, 99, 100, 8, 101, 102,
103, 80, 81, 84, 85, 104, 105, 9, 106, 41, 79, 80, 81, 82, 83, 41, 54, 107, 108,
59, 9, 99, 109, 22, 110, 111, 112, 113, 114, 115, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0]


Sentence:
 性能挺不错的，这么小个头这个性能蛮不错，装上后电脑复活了，原先卡死了，现在开机7秒左右非常快，这么大容量装游戏什么的美滋滋。MLC颗粒寿命有保证，看到TLC就讨厌，听说以后主流TLC，主打性价比的会是QLC寿命更低，所以MLC颗粒以后难买了，价格不知道得多贵啊[PAD][PAD][PAD][PAD][PAD][PAD]
[PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD]


Seq_len_list:
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 9, 18, 12, 19, 20,
21, 22, 23, 9, 24, 25, 26, 27, 15, 28, 9, 29, 30, 31, 32, 33, 34, 35, 36, 37, 9,
38, 39, 40, 41, 42, 8, 43, 44, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0]
[45, 46, 47, 22, 23, 8, 9, 48, 35, 28, 49, 50, 48, 49, 45, 46, 51, 22, 23, 9,
52, 53, 54, 55, 56, 57, 58, 59, 9, 60, 61, 62, 63, 59, 9, 33, 64, 18, 12, 65,
66, 67, 68, 69, 70, 71, 9, 48, 35, 72, 73, 74, 52, 75, 76, 34, 35, 8, 77, 78,
78, 44, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 9, 89, 90, 91, 80, 81, 92, 93,
94, 9, 95, 96, 41, 54, 11, 97, 91, 80, 81, 9, 11, 98, 45, 99, 100, 8, 101, 102]
```

```
[69, 70, 22, 23, 9, 71, 116, 117, 118, 119, 15, 71, 9, 120, 2, 32, 119, 21, 71,
9, 121, 97, 19, 20, 21, 69, 70, 71, 9, 122, 123, 16, 17, 9, 12, 124, 31, 86, 36,
37, 9, 125, 126, 127, 20, 128, 22, 23, 129, 9, 130, 131, 32, 132, 21, 86, 9, 61,
42, 133, 53, 134, 89, 89, 86, 31, 86, 135, 136, 36, 37, 9, 86, 36, 37, 137, 138,
139, 140, 129, 9, 16, 17, 8, 7, 121, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[55, 56, 141, 142, 15, 143, 9, 144, 145, 146, 147, 148, 149, 147, 9, 150, 128,
102, 151, 152, 146, 147, 44, 55, 56, 18, 12, 15, 71, 9, 153, 108, 154, 102, 48,
155, 9, 15, 156, 157, 158, 159, 160, 161, 161, 162, 9, 89, 89, 55, 163, 95, 164,
9, 165, 166, 15, 167, 168, 169, 170, 15, 171, 172, 92, 173, 174, 175, 72, 72, 8,
176, 177, 175, 44, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0]
[178, 52, 69, 70, 179, 77, 55, 56, 69, 70, 179, 180, 171, 172, 64, 48, 49, 99,
181, 108, 90, 48, 35, 182, 8, 55, 56, 183, 184, 59, 185, 24, 25, 69, 70, 97,
186, 121, 97, 69, 70, 71, 9, 187, 55, 32, 119, 188, 2, 125, 126, 15, 182, 189,
125, 184, 184, 8, 185, 190, 191, 2, 192, 193, 49, 194, 185, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[195, 195, 196, 9, 182, 197, 198, 9, 99, 197, 121, 77, 185, 199, 200, 189, 125,
9, 201, 202, 19, 20, 203, 204, 205, 9, 32, 119, 19, 20, 9, 106, 6, 121, 97, 206,
167, 207, 154, 15, 22, 23, 208, 209, 209, 28, 210, 211, 212, 213, 214, 134, 12,
201, 202, 209, 215, 91, 216, 55, 56, 134, 12, 217, 42, 218, 219, 220, 202, 209,
40, 221, 222, 223, 224, 80, 225, 226, 225, 209, 227, 147, 209, 192, 193, 216,
228, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[229, 230, 230, 118, 119, 21, 15, 71, 44, 231, 138, 92, 18, 12, 232, 59, 44,
233, 234, 32, 8, 102, 12, 235, 8, 44, 18, 12, 193, 236, 113, 66, 44, 150, 204,
205, 15, 72, 44, 41, 54, 237, 238, 239, 137, 240, 52, 129, 44, 241, 242, 230,
243, 244, 44, 12, 124, 102, 245, 246, 8, 44, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

实现过程

首先定义名为Tokenizerpro的新类，并定义对应的方法如decode，tokenize等。

之后在main函数中读取名为jd_comments的txt文档，并进行实例化对象的创建（命名为new）。

然后对new对象进行一系列的操作，输出第二条评论的编码后的数字序列，重新翻译后的字符串等，保存在名为output的txt文档中。

相关文件

作业相关的代码以及其他文件已上传至北航云盘和Github代码库