程设第五次作业

20377383 樊思涵

创建 Tokenizer

```python
class Tokenizer:
    def __init__(self,chars,coding='c',PAD=0):
        """
        输入将要需要操作的文本，完成词典的构建
        coding='w'按词构建
        coding='c'按字构建（默认）
        PAD默认为0
        将字典赋值给dic_chars
        """
        self.chars=chars
        self.coding=coding
        self.PAD=PAD
        dic={}
        dic['[PAD]'] = 0
        code_number = 1
        if coding == 'c':
            for char in chars:
                if char not in dic:
                    dic[char] = code_number
                    code_number += 1
        elif coding == 'w':
            lis_words=jieba.lcut(chars)
            for word in lis_words:
                if word not in dic:
                    dic[word] = code_number
                    code_number += 1
        self.dic_chars=dic
```

建立类变量 dic_chars 并赋值构建的词典

定义 tokenize 函数

```
31
32      def tokenize(self, sentence) -> list:
33          """
34          输入句子sentence
35          返回list_of_chars
36          """
37          list_of_chars=[]
38          if self.coding == 'c':
39              for char in sentence:
40                  list_of_chars.append(char)
41          if self.coding == 'w':
42              list_of_chars = jieba.lcut(sentence)
43          return list_of_chars
44
```

定义 encode 函数

```
45      def encode(self, list_of_chars):
46          """
47          输入字符(字或者词）的字符列表，返回转换后的数字列表 (tokens)。
48          """
49          tokens = []
50          for char in list_of_chars:
51              tokens.append(self.dic_chars[char])
52          return tokens
53
```

定义 trim 函数

```
54      def trim(self, tokens, seq_len):
55          """
56          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。
57          """
58          while len(tokens) < seq_len:
59              tokens.append(0)
60          if len(tokens) > seq_len:
61              tokens = tokens[:seq_len]
62          return tokens
63
```

定义 decode 函数

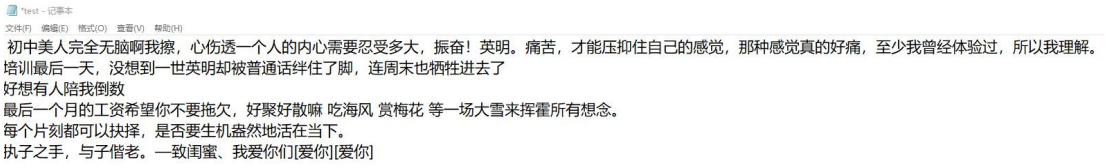
```
63
64      def decode(self, tokens):
65          """
66          将数字列表翻译回句子
67          """
68          for i in tokens:
69              for k,v in self.dic_chars.items():
70                  if i == v:
71                      print(k,end = '')
72          print()
73
```

定义 encode_all 的函数

```python
    def encode_all(self,seq_len) -> list:
        """
        返回所有文本(chars)的长度为seq_len的tokens
        """
        list_of_chars = self.tokenize(self.chars)
        tokens = self.encode(list_of_chars)
        lis_tokens = []
        num = len(tokens)
        for i in range(int(num / seq_len)):
            lis_tokens.append(tokens[seq_len * i:seq_len * i + seq_len])
        lis_tokens.append(self.trim(tokens[seq_len * int(num / seq_len):],seq_len))
        return lis_tokens

```

尝试类的实例化
测试用文本



类的实例化与类的方法

```python

with open(filename,'r',encoding='utf-8') as f:
    chars = f.read()
with open(filename,'r',encoding='utf-8') as f:
    string = f.readlines()
print(string[3])
T = Tokenizer(chars,'c')
list_of_chars = T.tokenize(string[3])
print(list_of_chars)
tokens = T.encode(list_of_chars)
print(tokens)
seq_len = 10
print('seq_len = %d'%seq_len)
tokens = T.trim(tokens,seq_len)
print(tokens)
T.decode(tokens)
print(T.encode_all(seq_len))
```

结果展示：

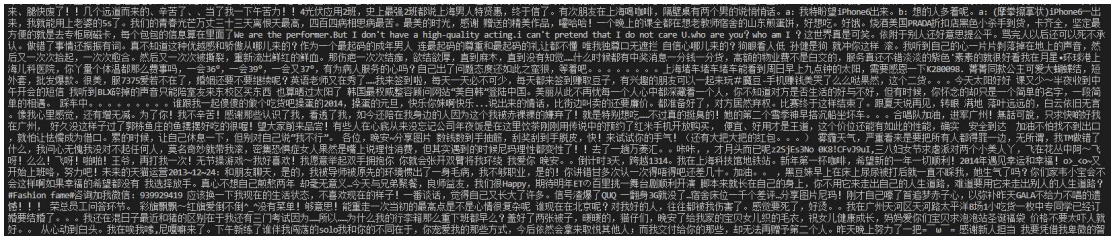
```
['最', '后', '一', '个', '月', '的', '工', '资', '希', '望', '你', '不', '要', '拖', '欠', '，', '好', '聚', '好', '散', '嘛', ' ', '吃', '海', '风', ' ', '赏', '梅', '花', ' ', '等', '一', '场', '大', '雪', '来', '挥', '霍', '所', '有', '想', '念', '。', '\n']
[62, 63, 17, 18, 89, 19, 90, 91, 92, 93, 94, 95, 22, 96, 97, 13, 47, 98, 47, 99, 100, 1, 101, 102, 103, 1, 104, 105, 106, 1, 107, 17, 108, 26, 109, 110, 111, 112, 55, 85, 66, 113, 32, 59]
seq_len = 10
[62, 63, 17, 18, 89, 19, 90, 91, 92, 93]
最后一个月的工资希望
[[1, 2, 3, 4, 5, 6, 7, 8, 9, 10], [11, 12, 13, 14, 15, 16, 17, 18, 5, 19], [20, 14, 21, 22, 23, 24, 25, 26, 13, 27], [28, 29, 30, 31, 32, 33, 34, 13, 35, 36], [37, 38, 39, 40, 41, 19, 42, 43, 13, 44], [45, 42, 43, 46, 19, 47, 33, 13, 48, 49], [11, 50, 51, 52, 53, 54, 13, 55, 56, 11], [57, 58, 32, 59, 60, 61, 62, 63, 17, 64], [13, 65, 66, 67, 17, 68, 30, 31, 69, 70], [71, 72, 73, 74, 39, 75, 76, 13, 77, 78], [79, 80, 81, 82, 83, 84, 75, 59, 47, 66], [85, 5, 86, 11, 87, 88, 59, 62, 63, 17], [18, 89, 19, 90, 91, 92, 93, 94, 95, 22], [96, 97, 13, 47, 98, 47, 99, 100, 1, 101], [102, 103, 1, 104, 105, 106, 1, 107, 17, 108], [26, 109, 110, 111, 112, 55, 85, 66, 113, 32], [59, 114, 18, 115, 116, 117, 118, 56, 119, 120], [13, 121, 122, 22, 123, 124, 125, 126, 127, 128], [129, 130, 131, 32, 59, 132, 133, 134, 135, 13], [136, 133, 137, 138, 32, 139, 140, 141, 142, 143], [11, 144, 94, 145, 146, 144, 94, 147, 146, 144], [94, 147, 0, 0, 0, 0, 0, 0, 0, 0]]
PS E:\code\py_code>
```

对目标文件进行分析处理

本次作业文件格式处理与 week3 微博数据清洗基本相同，调用之前已完成的函数并略加修改（删除这次不需要的表情包内容，本次实验数据没有重复项，不需要删除重复项操作）

```
 91 > def fread_document(filename) -> list:…
109
110 > def fdelete_repetition_txt(txt) -> list:…
120
121 > def fcut_txt(txt,n) -> list:…
134
135 > def fdelete_url_data(data,n) -> list:…
149
150 > def fclean_sentence(sentence,n) -> list:…
164
165   def main():
166       """
167       main函数
168       """
169       txt=fread_document(filename_0) #读取txt文件
170       m=len(txt)
171       #txt=fdelete_repetition_txt(txt) #删除重复项
172       n=len(txt) #记录项数
173       #print("处理前数据有%d项.\n处理重复数据后有%d项." %(m,n))
174       data=fcut_txt(txt,n) #简单切割内容、地址、时间
175       n=len(data)#更新项数
176       #print(n)
177       sentence = fdelete_url_data(data,n) #简单分词处理
178       sentence = fclean_sentence(sentence,n) #正则表达式降噪
179       #print(sentence)
```

处理结果展示



将上述文本建立按词编码和按字编码两种类，并计算原文本规模与编码规模

```
184       T_c = Tokenizer(chars,'c')
185       print("文本总长度为%d"%T_c.len_all)
186       print('按字编码的编码规模为%d'%len(T_c.dic_chars))
187       T_w = Tokenizer(chars,'w')
188       print("文本总词数为%d"%T_w.len_all)
189       print('按词编码的编码规模为%d'%len(T_w.dic_chars))
```

结果展示如下：

```
PS E:\code\py_code> python -u "e:\code\py_code\week5\week5.py"
----------正在导入数据----------
----------数据导入完成----------
文本总长度为49964824
按字编码的编码规模为12160
Building prefix dict from the default dictionary ...
Loading model from cache C:\Users\LF\AppData\Local\Temp\jieba.cache
Loading model cost 0.580 seconds.
Prefix dict has been built successfully.
文本总词数为32051333
按词编码的编码规模为522580
```

按字编码：
原文本字数长度为五千万字，使用 tokenizer 后的编码规模为 12000，为原文本量的千分之二(2.434×10^{-4})

按词编码：
原文本总词数为三千万，使用 tokenizer 后的编码规模为 522580，为原词数的百分之二(0.0163)

tokenizer 方法与第二周作业中的 one-hot 方法编码之间的区别和优劣

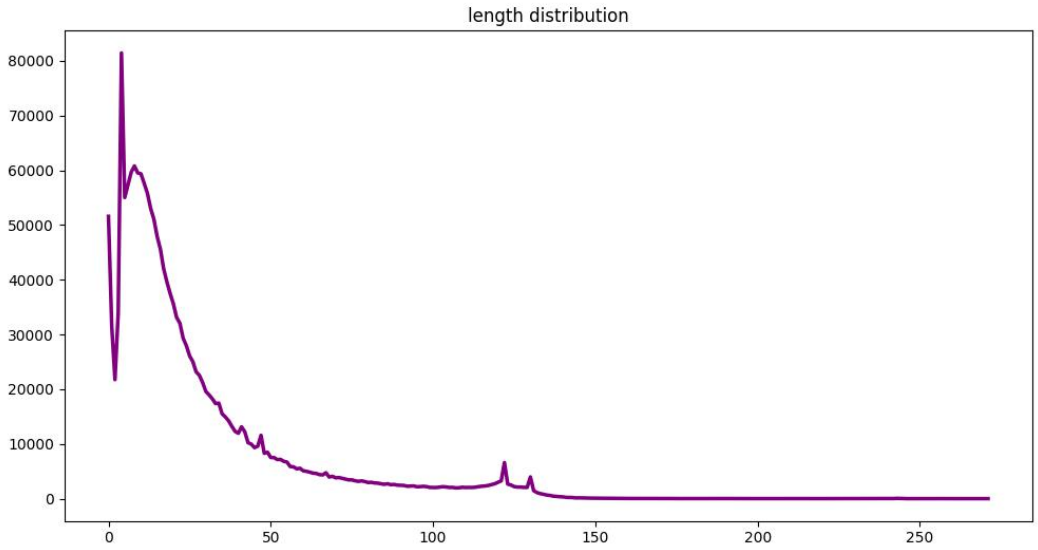
区别：
第二周作业中的 one-hot 方法编码是将所有的热点词创建向量，再获得对应文本的向量状态。
本次的 tokenizer 方法是先通过整个文本获得编码字典，再通过字典的法则将对应的文本进行编码。

优劣：
第二周作业中的 one-hot 方法解决了分类器不好处理离散数据的问题，分析向量特征时较为方便，缺点则是它不考虑词与词之间的顺序，无法通过向量还原文本，并且得到的特征是离散稀疏的。
本次作业中的 tokenizer 方法在编码，解码的过程中不会丢失文本的信息，词（字）与词（字）间的顺序仍然得到保留，缺点则是信息的特征表现不明显。

确定一个合适的 seq_len(以 coding='c'为例)

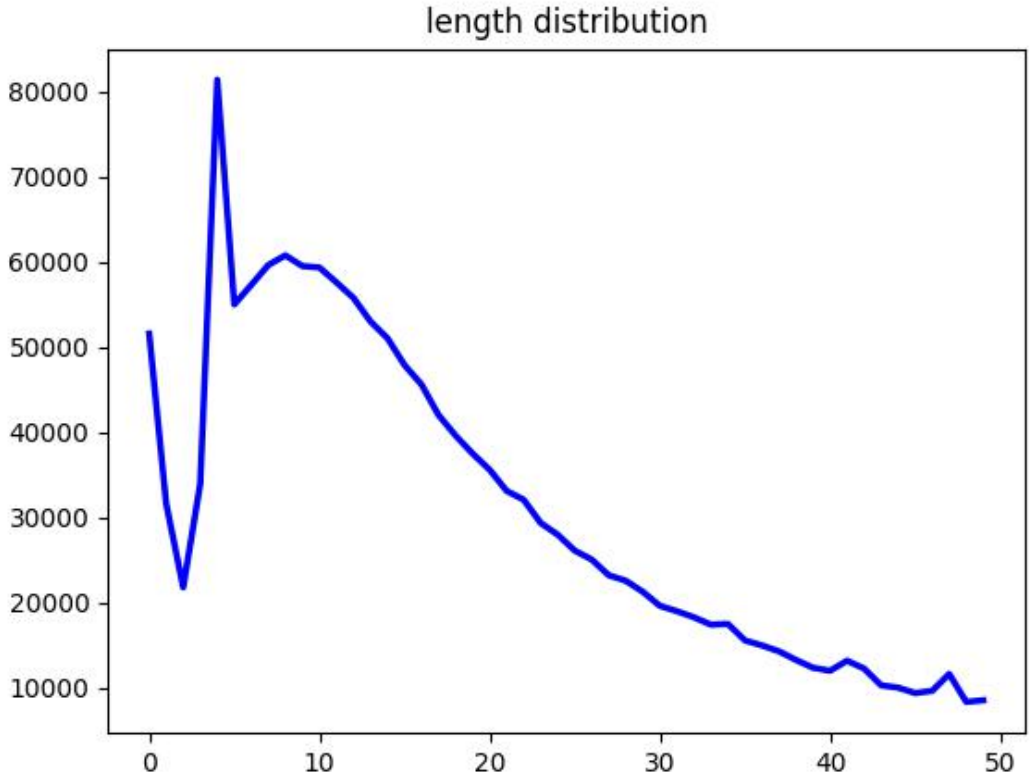
```
207     lis_len=[]
208     for i in sentence:
209         lis_len.append(len(i))
210     max_len=max(lis_len)
211     dis_len=[0 for i in range(max_len+1)]
212     for i in lis_len:
213         dis_len[i] += 1
214     print(max_len)
215     print(dis_len)
216     x=[i for i in range(max_len+1)]
217     y=dis_len
218     plt.plot(x,y,color="purple",linewidth=2.5)
219     plt.title("length distribution")
220     plt.show()
```

分布图如下



进一步观察长度为 0~50 的长度分布图

```
221     x=[i for i in range(50)]
222     y=dis_len[:50]
223     plt.plot(x,y,color="blue",linewidth=2.5)
224     plt.title("length distribution")
225     plt.show()
```



其中峰值点长度为 4，故此为可考虑的 **seq_len**
但在这种情况下大半的句子将被切割。
除此结果外，还可考虑长度均值，或 75%分位点(即百分之 75 的句子将不会被切割)的 seq_len

计算长度均值

```
237     seq_len = round(len(chars)/len(sentence))
238     print(seq_len)
```

输出长度为 27

下面计算 75%分位点的句子长度

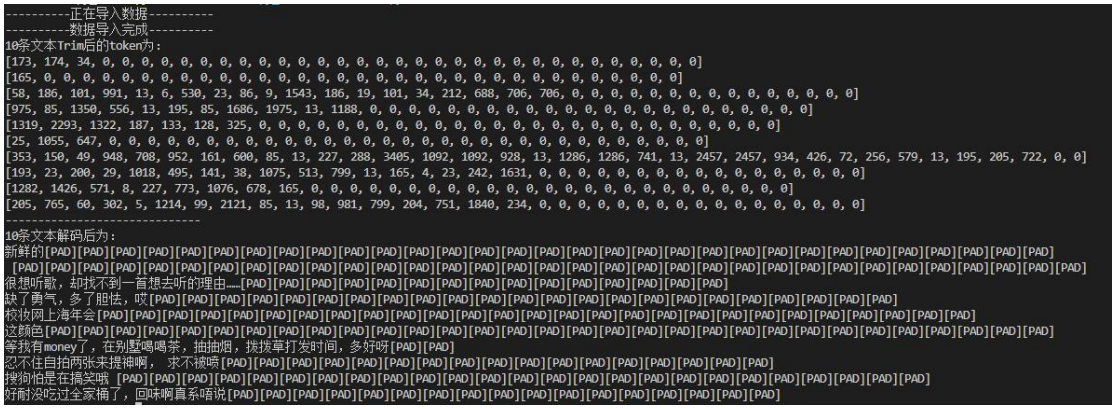
```
230     count = 0
231     lis_i = 0
232     while(count < n_txt * 0.75):
233         count += dis_len[lis_i]
234         lis_i += 1
235     print(lis_i-1)
```

输出长度为 34

三种 **seq_len** 长度各有优劣，下面我们采用 75%分位点的 **seq_len** 随机输出 10 个句子的按字编码的 token 并将其解码。

```
192     T_c = Tokenizer(chars,'c')
193     lis_token=[]
194     seq_len = 34
195     for i in range(10):
196         number = random.randint(0,len(sentence)-1)
197         lis_of_chars = T_c.tokenize(sentence[number])
198         tokens = T_c.encode(lis_of_chars)
199         tokens = T_c.trim(tokens,seq_len)
200         lis_token.append(tokens)
201     print("10条文本Trim后的token为：")
202     for i in lis_token:
203         print(i)
204     print('------------------------------')
205     print("10条文本解码后为：")
206     for i in lis_token:
207         T_c.decode(i)
```

结果展示：


----------正在导入数据----------
----------数据导入完成----------
10条文本Trim后的token为：
[173, 174, 34, 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]
[165, 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]
[58, 186, 101, 991, 13, 6, 530, 23, 86, 9, 1543, 186, 19, 101, 34, 212, 688, 706, 706, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[975, 85, 1350, 556, 13, 195, 85, 1686, 1975, 13, 1188, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[1319, 2293, 1322, 187, 133, 128, 325, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[25, 1055, 647, 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]
[353, 150, 49, 948, 708, 952, 161, 600, 85, 13, 227, 288, 3405, 1092, 1092, 928, 13, 1286, 1286, 741, 13, 2457, 2457, 934, 426, 72, 256, 579, 13, 195, 205, 722, 0, 0]
[193, 23, 200, 29, 1018, 495, 141, 38, 1075, 513, 799, 13, 165, 4, 23, 242, 1631, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[1282, 1426, 571, 8, 227, 773, 1076, 678, 165, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[205, 765, 60, 302, 5, 1214, 99, 2121, 85, 13, 98, 981, 799, 204, 751, 1840, 234, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

10条文本解码后为：
新鲜的[PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD]
[PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD]
很想听歌，却找不到一首想去听的理由……[PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD]
缺了勇气，多了胆怯，哎[PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD]
校妆网上海年会[PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD]
这颜色[PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD]
等我有money了，在别墅喝喝茶，抽抽烟，拨拨草打发时间，多好呀[PAD][PAD]
忍不住自拍两张来提神啊， 求不被喷[PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD]
搜狗怕是在搞笑哦 [PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD]
好耐没吃过全家桶了， 回味啊真系唔说[PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD]