第5周作业 文本处理

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，在本次作业提供的文本数据集上进行测试，同时讨论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。
- 代码结构：
  a. __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。
  b. tokenize(self, sentence) 输入一句话，返回分词(字）后的字符列表(list_of_chars)。
  c. 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)。
  d. 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。
  e. 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]'。
  f. encode_all(self，seq_len) 返回所有文本(chars)的长度为seq_len的tokens。
- 提示：seq_len是句子的长度，实际任务中**如何确定一个合适的长度**，请以本次作业中的文本数据为例，通过文本的长度分布来进行观察和讨论。
- （附加）尝试使用BERT预训练模型（https://huggingface.co/bert-base-chinese）中提供的tokenizer对本次样本中的文本进行编码，并使用其预训练模型得到句子的表征向量。抽取本次数据中的一条文本，并查找其相似文本，并与第二周中使用word2wec得到的结果相比较，分析两者优劣。