1、源代码

'''

模型：bigram

数据平滑方法：拉普拉斯平滑

'''

import re

def dataPreproessing(src):  *#进行数据预处理，返回词句列表*

    ans = []

    with open(src, 'r', encoding="utf-8") as f:

        datas = f.readlines()

        for txt in datas:

            if txt == "\n":

                continue

*# 正则表达式匹配，去除相应的字符*

            pattern = r',|\.|/|;|\'|`|\[|\]|<|>|\?|:|"|\{|\}|\~|!|@|#|\$|%|\^|&|\(|\)|-|=|\\_|\+|，|。|、|；|‘|’|【|】|·|！| |…|（|）|“|”|nbsp|\n|\u3000|[0-9]|[a-z]|[A-Z]|《|》|——|？|/'

            lst = re.split(pattern, txt)

            lst = list(filter(lambda x: x != '', lst))

            if lst != []:

                ans += lst

    return ans

def bigramCounting(ans: list) -> dict:      *# 第一维是w\_{i-1}的，第二维是w\_i的*

*# 统计出现的字*

    s = set()

    for txt in ans:

        for ch in txt:

            s.add(ch)

    s.add("<BOS>")

    s.add("<EOS>")

    dictionary = {}

*# 创建词典*

    for ch in s:

        dictionary[ch] = {}

    for ch in s:

        for ch\_sub in s:

            dictionary[ch][ch\_sub] = 0

*# 统计2-gram对*

    for txt in ans:

        for idx, ch in enumerate(txt):

            if idx + 1 == len(txt):     *# 到达最后一个字，不再统计*

                break

            else:

                nxt = txt[idx + 1]

            dictionary[ch][nxt] += 1

    return dictionary

def laplaceSmoothing(countDictionary: dict):    *# 拉普拉斯平滑*

    for key, val in countDictionary.items():

        for key\_sub, val\_sub in val.items():

            countDictionary[key][key\_sub] += 1

    return countDictionary

def getProbDict(countDictionary: dict):     *# 第一维是条件w\_{i-1}，第二维是w\_i，统计的是条件概率P(w\_i|W\_{i-1})*

    for w\_i\_1, dic in countDictionary.items():

        tot = 0

        for w\_i, val in dic.items():

            tot += val

        if tot == 0:

            continue

        for w\_i, val in dic.items():

            countDictionary[w\_i\_1][w\_i] = val / tot

    return countDictionary

def bigramPred(sentence: str, dictionary: dict, topK: int = 5): *# 2-gram模型的预测*

    w\_i\_1 = sentence[-1]        *# 因为使用的2-gram模型，下一个字的预测只与句子的最后一个字有关*

    lst = []

    for key, val in dictionary[w\_i\_1].items():

        lst.append((key, val))

    lst.sort(key = lambda x:x[1], reverse = True)

    num = min(len(lst), topK)

    return lst[0:num]

def formatPrint(sentence, ans):     *# 输出结果*

    print(sentence, "推荐列表及概率为：")

    for ch, prob in ans:

        sen = sentence + ch

        print("\t{:s}: {:2%}".format(sen, prob))

    print("")

def main():     *# main函数*

    ans = dataPreproessing("news.txt")

    dictionary = bigramCounting(ans)

    dictionary = laplaceSmoothing(dictionary)

    probDict = getProbDict(dictionary)

    while True:

        print("请输入您需要预测的字序列，按[Ctrl+C]退出：")

        s = input()

        try:

            lst = bigramPred(s, probDict)

            formatPrint(s, lst)

        except Exception as e:

            print("您输入的字符在语料训练库中不存在，请重新输入！\n")

main()

2、程序运行截图

