大数据原理与技术 平时作业 2

林宇浩 21311274

一、算法说明

1、Shingling

Shingling 是一种文本数据处理技术，通常用于文档相似性比较和搜索引擎索引等领域。它的基本思想是将文档或文本片段转换为固定长度的子串集合，这些子串通常称为“shingles”或“k-shingles”。然后，可以使用这些 shingles 来构建文本的特征向量，通常采用 0 和 1 来表示某个 shingle 是否在文本中出现。Shingling 可以用于文本相似度计算、文本分类等任务中。具体来说，Shingling 过程包括以下步骤：

1. 切分文本：首先，将文档或文本片段切分为连续的短序列，通常是单词或字符。
2. 构建 shingles：然后，以固定长度 k 滑动窗口的方式遍历文本，并将每个窗口内的序列组合成一个 shingle。
3. 去重：最后，对所有生成的 shingles 进行去重操作，以保证每个 shingle 的唯一性。

下面是一个例子，假设我们有两个文本：

文档 A：The quick brown fox jumps over the lazy dog.

文档 B：A quick brown dog jumps over the lazy fox.

我们将以单词为单位，使用长度为 3 的 shingles。在这种情况下，我们的 shingles 可能包括：

文档 A 的 shingles：{"the quick brown", "quick brown fox", "brown fox jumps", "fox jumps over", "jumps over the", "over the lazy", "the lazy dog"}

文档 B 的 shingles :{"a quick brown", "quick brown dog", "brown dog jumps", "dog jumps over", "jumps over the", "over the lazy", "the lazy fox"}

2、MinHashing

MinHashing 是一种用于快速近似计算集合相似性的技术，通常用于大规模文本数据或网络数据中。它通过对集合进行哈希函数映射，从而将大型集合压缩成固定长度的签名向量，然后使用这些签名向量进行比较，从而估算集合之间的相似性。

MinHashing 的基本思想是利用一组哈希函数来找到每个集合的“最小”元素，然后使用这些最小元素的哈希值来构建签名。通过使用多个哈希函数，可以生成多个哈希值，从而构建出一个签名矩阵。最终，将这些哈希值中的最小值作为集合的 MinHash 签名。

3、Locality Sensitive Hashing (LSH)

局部敏感哈希（Locality Sensitive Hashing，LSH）是一种用于高维数据近似最近邻搜索的技术。它的主要思想是将数据点哈希到哈希空间中，使得相似的数据点在哈希空间中具有高概率映射到同一个哈希桶中，从而在哈希空间中可以高效地找到相似的数据点。LSH 算法的主要步骤包括：

哈希函数的选择：LSH 算法中哈希函数可以将数据点映射到哈希空间中的某个桶。哈希函数的选择需要满足局部敏感性的要求，即相似的数据点被映射到相同的桶的概率较高。

查询：当需要找到某个查询点的近邻时，将查询点通过相同的哈希函数映射到哈希空间中，并在哈希表中搜索相似的数据点。由于相似的数据点有较高的概率被映射到相同的桶中，因此可以在桶中搜索附近的数据点来找到近邻。

候选筛选：由于哈希函数的敏感性有限，可能存在一些不相似的数据点也被映射到相同的桶中。因此，在找到候选近邻后，需要对候选进行进一步的筛选，以确保找到的近邻是真正相似的。

4、总结：

Shingling 将文本处理成 one-hot 向量，代表字符串的集合。

MinHashing 将 one-hot 向量处理为签名，用签名计算 Jaccard 相似性是用 one-hot 向量计算 Jaccard 相似性的一个估计。

LSH 将相似的签名映射到同一个桶中。

二、展示

1、数据集：

实验数据如下所示，使用的数据集为一个中文购物评论数据集，总共有 10 个类，类标签表示正面评论或负面评论。

```
1   cat,label,review
2   书籍,1,@做父母一定要有刘墉这样的心态，不断地学习，不断地进步，不断地给自己补充新鲜血液，让自己保持一颗年轻的心。我想，这是他能很好的和孩子沟通的一个重要因素。i
3   书籍,1,作者真有英国人严谨的风格，提出观点、进行论述论证，尽管本人对物理学了解不深，但是仍然能感受到真理的火花。整本书的结构颇有特点，从当时（本书写于八十年代）
4   书籍,1,作者长篇大论借用详细报告数据处理工作和计算结果支持其新观点。为什么荷兰曾经县有欧洲最高的生产率？为什么在文化上有着深刻纽带关系的中国和日本却在经济发展上
5   书籍,1,作者在战几时之前用了＂拥抱＂令人叫绝．日本如果没有战败，就有会有美军的占领，没胡官僚主义的延续，没有战后的民发反思，没有～，就不会让日本成为一个经济强国
6   书籍,1,作者在少年时即喜阅读，能看出他精读了无数经典，因而他有一个庞大的内心世界。他的作品最难能可贵的有两点，一是他的理科知识不错，虽不能媲及罗素，但与理科知识
7   书籍,1,作者有一种专业的谨慎，若能有幸学习原版也许会更好，简体版的书中的印刷错误比较多，影响学者理解，全书结构简单，但内容详实，学起来如鱼得水非常轻松。这只是一
8   书籍,1,作者用诗一样的语言把如水般清澈透明的思想娓娓道来，像一个经验丰富的智慧老人为我们解开一个又一个人生的难题，让我们超越自发的自己和世俗的观念，引导着我们走
9   书籍,1,作者提出了一种工作和生活的方式，作为咨询界的元老，不仅能提出理念，而且能够身体力行地实践，并且提出具体的步骤，展现的是一种自然的生活态度，一种劳役结合、
10  书籍,1,作者妙语连珠，将整个60-70年代用层出不穷的摇滚巨星与自身故事紧紧相连什么是乡愁？什么是摇滚？那是一种特立独行看似高傲实则温暖的姿态那是一种绝世独立下深深
11  书籍,1,作者逻辑严密，一气呵成。没有一句废话，深入浅出，循循善诱，环环相扣。让平日里看到指标图释就头疼的我竟然大有拨云见日，醍醐灌顶之感。果不出所料，占豪的黄金
12  书籍,1,作者力从马克思注意经济学角度来剖析当代中国经济细心的人会发现中国近20年来的一些政策措施在何新先生的文章里找到蛛丝马迹作者早在2001年就预言十年内世界会爆发
```

```
1   cat,label,review
27195   洗发水,1,收到了，还没用，谢谢厂家，如果是正品就比外面超市优惠太多，还送吹风，还送了两瓶一百毫升的，真好，下次再光临
27196   洗发水,1,最近用了几次这个洗发水，还可以，京东做活动送货也慢了啊
27197   洗发水,1,收到货了、只有3个字形容（高大上）！不到*得到这么多的东西！真是不错、不知道好不好用、用了再追加
27198   洗发水,1,618买的，送了很多赠品，比实体店划算，自从接触了这个品牌，一直都用它了，效果真不错，头发明显不掉了，也不干燥，特别滋润，有光泽度，我身边的亲
27199   洗发水,1,是正品，很好的，这次搞活动动买的，价格便宜不错，相信京东品质，无可挑剔！。
27200   洗发水,1,非常超值，价格还比现在优惠，然后包装很好，次日达非常方便。
27201   洗发水,1,感觉还不错，和之前买的一样，之前买的89一瓶，6.18搞活动两瓶才95还送了一小瓶，还送了吹风机
27202   洗发水,1,凑单买的，比海飞丝便宜量足！使用起来效果还行，没有广告那么夸张
27203   洗发水,1,泡沫少，香味淡，包装小，没想象中好，就是广告响
27204   洗发水,1,清扬洗发水家里一直在用，个人感觉很适合，用完后清爽自然。信赖京东自营京东快递准时快速，方便省心！快递员服务态度一流。
```

```
1   cat,label,review
38948   衣服,1,"质量非常好，与卖家描述的完全一致，非常满意,真的很喜欢，完全超出期望值，发货速度非常快，包装非常仔细、严实，物流公司服务态度很好，运送速
38949   衣服,1,收到货了，说一下个人看法给大家做个参考。首先款式和我想的差不多，跟图片上也一致，属于比较年轻的款，所以年龄上要有所考虑。再者这款裤子的码
38950   衣服,1,裤子质量很不错，物流也很好，值得推荐的店铺，值得购买
38951   衣服,1,裤子很满意，买给弟弟穿的，弟弟很喜欢，尺码，款式都很好，质量也很棒，很喜欢，会一直关注的。
38952   衣服,1,裤子质量特别好，穿在腿上特别舒服。物美价廉，总之一次不错的购物，下次还来他家购物。
38953   衣服,1,卖家服务态度特别好，超耐心，裤子版式也好，料子更是赞得不行，自己很喜欢，朋友也很喜欢，说也要买一条，真的好赞赞赞赞赞赞赞赞！
38954   衣服,1,收到了，这么优惠的价格收到质量不错的还挺满意的，而且都没有线头，老公很喜欢。
```

```
1   cat,label,review
60031   酒店,0,坦率地说，我认为云天海湾酒店在面积上有欺诈行为。酒店广告说，标准间B面积为33-37平方米，标准间A为44-47平方米，但7月17日我预订并入住了2号
60032   酒店,0,酒店时间太久，设施都很旧，空调整宿嗡嗡的响。这次最让人不满意的是入住后第二天晚上回到房间，看到房间内酒水消费单上莫名的被记上消费了一瓶35
60033   酒店,0,承德历史和自然风景很美，但饭店服务方面、、不敢恭维住宿：1、房间 标间设施简陋但还算干净，可是房间总是有一股刺鼻的家具气味；另外房间木门变
60034   酒店,0,相对于价格，酒店的硬件和服务实在有很大欠缺。首先，预定客房的时候说明不够清楚，酒店仅有很少数几间房间带窗户和双床，轮到我仅有无窗大床房，
60035   酒店,0,"这是我住过最差最差的酒店,夜里十一点服务员们在楼道大声喧哗聊天,其中两个服务员为谁干的活多活少还吵了起来,早上六点服务员就敲门要求搞卫生,
60036   酒店,0,房间设施相当差，马桶圈居然小了一大圈，浴缸的水直接排到卫生间的下水道，洗个澡弄的卫生间里全是水，门卡居然也没有，说是给交警大队队长了，进
60037   酒店,0,我是早上7点的国际航班，但是班车最早只有5点半才有，而且这里没法定出租车，只能叫朋友4点半开车过来送，太晕了。酒店就不能给安排个小车接送有
60038   酒店,0,位置不是很好，离泰山脚下比较远。房间3楼不靠街道的那面较好，虽然后面有市场，但是双窗，关闭后噪音还好。靠街道面的噪音很大，特别是二楼。附近
60039   酒店,0,前台服务太慢，很多住店客人不自居排队，服务员也不加制止。房间的空调味道太难闻，恶心。班车服务还是很不错的，准时。
```

实验的数据配置为，选取一个类作为主类，然后从其他类中抽取一些评论掺杂在主类中，我们的任务是为主类评论找到相似的评论，即找到同样属于主类的评论。由于本人的轻薄本电脑性能有限，主类评论选取了 70 条作为正样本，其他类的评论选取了 40 条作为负样本，共 150 条样本进行测试。每轮实验中数据均控制不变。

2、参数和符号说明：

k	Shingling 中滑动窗口的长度
M	signature matrix 的行数
b	band 的数量
r	每个 band 的行数，r=M/b
P	两个签名（signature matrix 的列）至少有一个 band 相同的概率，即两个签名为候选列对的概率，$P = 1 - (1 - t^r)^b$，t 为两个签名之间的相似性

3、实验结果：

实验结果详解下面的各表格，每个表格中横坐标代表 r 的值，纵坐标代表 b 的值
下表中表项的格式为：**(真阳性数量,假阳性数量) (真阴性数量,假阴性数量)**
真阳性指被判断为相似的主类评论
假阳性指被误判为相似的其他类评论
真阴性指被判断为不相似的其他类评论
假阴性指被误判为不相似的主类评论

k=1

r\b	10	50	100	500	1000
1	(39,20) (20,30)	(65,35) (5,4)	(65,35) (5,4)	(65,37) (3,4)	(65,37) (3,4)
2	(9,0) (40,60)	(27,1) (39,42)	(30,4) (36,39)	(62,30) (10,7)	(62,28) (12,7)
3	(5,0) (40,64)	(11,0) (40,58)	(18,1) (39,51)	(31,3) (37,38)	(38,3) (37,31)

k=2

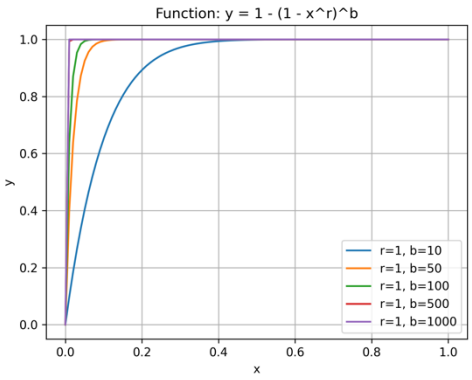
r\b	10	50	100	500	1000
1	(13,2) (38,56)	(33,12) (28,36)	(33,9) (31,36)	(38,14) (26,31)	(38,15) (25,31)
2	(2,0) (40,67)	(4,0) (40,65)	(7,2) (38,62)	(21,1) (39,48)	(29,2) (38,40)
3	(0,0) (40,69)	(1,0) (40,68)	(3,0) (40,66)	(5,0) (40,64)	(8,2) (38,61)

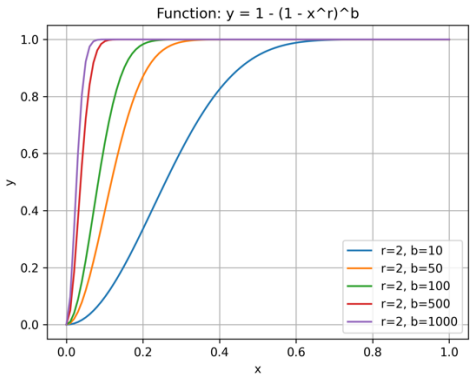
k=3

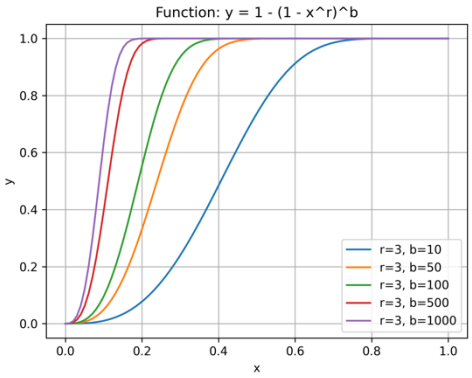
r\b	10	50	100	500	1000
1	(2,0) (40,67)	(4,1) (39,65)	(4,0) (40,65)	(4,1) (39,65)	(4,1) (39,65)
2	(0,0) (40,69)	(0,0) (40,69)	(0,0) (40,69)	(3,0) (40,66)	(3,0) (40,66)
3	(0,0) (40,69)	(0,0) (40,69)	(0,0) (40,69)	(0,0) (40,69)	(0,0) (40,69)

综合上面 3 个表格，可以看出随着 band 行数 r 值的增加，真阳性数量在减少，真阳性占总阳性的比例在上升。主要的原因是 r 越大，一个 band 中的行数就越多，两个样本要判断为相似需要有更多的行数一样，从而被判断为相似的概率更小，导致真阳性的数量在减少，负样本也更不容易被误判为相似。还可以看到，随着 b 的上升，真阳性和假阳性的数量在增加。主要原因是 b 越大，band 数量越多，能够判断为相似的机会更多，从而阳性结果的数量增加。对于 k，可以看到随着 k 的上升，阳性结果的数量在减少，主要原因由于数据结构所决定的，由于评论数据并不是特别长，较少评论之间会有三个一样的文字，导致 k 大于等于 3 之后大部分样本相互之间都无法匹配。

下面的三幅图展示了实验中所用的几种参数设置所对应的函数，按照 r 的不同绘制在了不同的图中。







可以看到，b 的值越大，函数中部的分界线越垂直，从而判决边界更加鲜明。对于参数 r，可以看到 r 的值越大，分界线的位置越靠近图像右侧，即对相似情况的要求更高。我们实验中两条评之间的相似内容较少，所以可以看到我们所用的几个函数的分界线都靠近图像左侧，即只要达到较低的相似度即可判断两个评论之间是相似的。b 的值越大，函数能够更清晰地进行判断，但是较大的 b 值将导致计算数据的成倍增加，带来耗时的大幅增加，这是需要权衡的。

三、代码说明

下面是代码的具体实现。首先，通过预处理文本函数 preprocess_text()去除文本中的标点符号和多余的空白字符，并将文本转换为小写形式。然后，通过 shingling()函数将文本切分为长度为 k 的 shingles，将 shingles 转换为 one-hot 向量，并使用 MinHash 处理生成签名矩阵。接下来，利用 LSH 算法将文档分配到不同的桶中，以加速相似度搜索。在处理 CSV 文件时，首先根据指定的类别列表选择特定类别的数据，然后对每个评论进行 shingling 和转换为 one-hot 向量，进行 MinHash 处理生成签名矩阵，并使用 LSH 算法进行相似度搜索，最后输出相似文档的统计结果。通过调整参数值，可以对不同的参数组合进行尝试，以寻找最佳的近似相似度搜索结果。具体详解详见下方代码注释：

```
import pandas as pd
import re
import numpy as np
from collections import defaultdict
import random

# 预处理文本：去除标点符号、多余空白字符，转换为小写
```

```
def preprocess_text(text):
    # 去除标点符号和文章格式
    text = re.sub(r'[^\w\s]', '', text)
    # 去除多余空白字符
    text = re.sub(r'\s+', '', text)
    # 转换为小写
    text = text.lower()
    return text

# 切分成 k 个字符的 shingles
def shingling(text, k):
    # 预处理文本
    text = preprocess_text(text)

    # 切分成 k 个字符的 shingles
    shingles = set()
    for i in range(len(text) - k + 1):
        shingles.add(text[i:i+k])

    if len(shingles) == 0:
        shingles.add(text)

    return shingles

# 将 shingles 转换为 one-hot 向量
def convert_to_one_hot(shingles, shingle_index):
    one_hot_vector = np.zeros(len(shingle_index), dtype=int)
    for shingle in shingles:
        index = shingle_index[shingle]
        one_hot_vector[index] = 1
    return one_hot_vector

# 判断矩阵是否有零行
# def has_zero_row(matrix):
#     for row in matrix:
#         if all(element == 0 for element in row):
#             return True
#     return False

# MinHash 处理
def min_hashing(one_hot_matrix, num_hashes):

    # print(has_zero_row(one_hot_matrix))

    set_num = one_hot_matrix.shape[1]  # 集合中元素的数量
    sample_num = one_hot_matrix.shape[0]  # 数据样本的数量

    unordered_matrix = []  # 样本数*集合长度
    for _ in range(num_hashes): # 生成乱序矩阵
        unordered_list = list(range(1, set_num + 1))
        random.shuffle(unordered_list)
        unordered_matrix.append(unordered_list)

    # print(unordered_matrix)

    # 初始化签名矩阵
    signature_matrix = np.full((num_hashes, sample_num), set_num+1, dtype=int)

    for i in range(sample_num):  # 遍历每个数据
        for j in range(num_hashes):  # 遍历每个哈希函数
            for k in range(set_num):  # 遍历哈希函数中每个值
                if unordered_matrix[j][k] < signature_matrix[j][i] and one_hot_matrix[i][k]
== 1:
                    signature_matrix[j][i] = unordered_matrix[j][k]

    # print(signature_matrix)

    return signature_matrix


def LSH(signature_matrix, bands, rows):

    # print(signature_matrix)

    signature_len = signature_matrix.shape[0]  # 获取签名矩阵的行数（即签名长度）
    doc_num = signature_matrix.shape[1]  # 获取签名矩阵的列数（即样本数量）
```

```
    # 初始化桶，以便于按照 band_signature 分类存储文档索引
    buckets = {}

    for b in range(bands):  # 遍历每个 band
        buckets[b] = defaultdict(list)
        for c in range(doc_num):  # 遍历每个文档
            # 计算当前 band 的起始和结束位置
            start = b * rows
            end = min(start + rows, signature_len)

            # 获取当前 band 中文档的签名，并将其转换为元组以便作为字典的键
            band_signature = tuple(signature_matrix[start:end, c])
            # print(band_signature)

            # 将当前文档索引存入相应的桶中
            buckets[b][band_signature].append(c)

    return buckets

# 处理 CSV 文件并执行 LSH 算法
def process_csv(csv_file_path, k, num_hashes, bands, rows):
    # 读取 CSV 文件
    df = pd.read_csv(csv_file_path)

    # 选择的类别列表
    selected_cats = ['蒙牛']
    hinder_cats = ['计算机', '书籍', '洗发水', '衣服']

    selected_cats = ['平板']
    hinder_cats = ['手机', '计算机']

    positive_data_num = 0
    negative_data_num = 0

    # 根据所选类别过滤数据
    selected_data = []
    for category in selected_cats:
        category_data = df[df['cat'] == category].head(70)
        selected_data.append(category_data)
        positive_data_num += len(category_data)

    for category in hinder_cats:
        category_data = df[df['cat'] == category].head(20)
        selected_data.append(category_data)
        negative_data_num += len(category_data)
    selected_df = pd.concat(selected_data)

    # 获取评论列
    texts = selected_df['review'].tolist()
    # labels = selected_df['cat'].tolist()
    # labels = [1 if x == "酒店" else x for x in labels]

    shingle_index = {}
    all_shingles = set()

    # 1. 对每个评论进行 shingling，得到每个评论对应的字符串集合
    all_sets = []
    for text in texts:
        shingles = shingling(text, k)
        # print(shingles)
        all_sets.append(shingles)
        all_shingles.update(shingles)

    # 2. 整合所有评论，将各个字符串集合转换为 one-hot 向量
    shingle_index = {shingle: i for i, shingle in enumerate(all_shingles)}
    one_hot_matrix = np.zeros((len(all_sets), len(all_shingles)), dtype=int)
    for i, shingles in enumerate(all_sets):
        one_hot_matrix[i] = convert_to_one_hot(shingles, shingle_index)

    # 3. 将各个 one-hot 向量进行 MinHashing 处理，得到签名矩阵
    signature_matrix = min_hashing(one_hot_matrix, num_hashes)

    # 4. 使用签名矩阵实现 Locality Sensitive Hashing
    buckets = LSH(signature_matrix, bands, rows)

    data_similar_to_0 = set()
    for index, one_hash in buckets.items():  # 遍历每个哈希函数，即每个 band
```

```
        for key, bucket in one_hash.items():  # 遍历哈希函数中的每个桶
            if 0 in bucket:
                data_similar_to_0.update(bucket)

    right_classify = sum(1 for x in data_similar_to_0 if x < positive_data_num)
    right_classify = right_classify-1  # 减去 0 号数据自己

    # 统计结果
    true_pos = right_classify
    false_pos = (len(data_similar_to_0)-1)-right_classify  # 集合总长也去除 0 号自己
    true_neg = negative_data_num-false_pos
    false_neg = (positive_data_num-1)-true_pos

    # print("真阳性:", true_pos)  # 真阳性
    # print("假阳性:", false_pos)  # 假阳性
    # print("真阴性:", true_neg)  # 真阴性
    # print("假阴性:", false_neg)  # 假阴性

    print(f"({true_pos},{false_pos}) ({true_neg},{false_neg})")

    return buckets


# 参数设置
# k = 2  # shingle 的长度
# num_hashes = 50  # MinHash 的哈希函数个数
# bands = 10  # LSH 中的 band 数
# rows = 5  # 每个 band 中的行数

# CSV 文件路径
csv_file_path = 'online_shopping_10_cats.csv'

for k in [1, 2, 3]:
    for r in [1, 2, 3]:
        for b in [10,50,100,500,1000]:
            print(f"k={k},r={r},b={b}")
            process_csv(csv_file_path, k, r*b, b, r)  # 处理 CSV 文件数据并执行 LSH 算法
```