实验设计文档

1 系统功能描述

1.1 说明：这是一套基于实例的图像检索系统，在界面选定一张实例图像，此应用程序将按评估排名的顺序从数据库返回包含相同实例的 10 张图像。

1.2 使用须知：

1.2.1 Python 环境配置：依赖保存在 requirement.txt 里，建议创建并将依赖安装在虚拟环境 venv，可以保证程序能正确运行。

1.2.2 数据集配置：请将所有 5K 张图片置于 static/images 目录下。

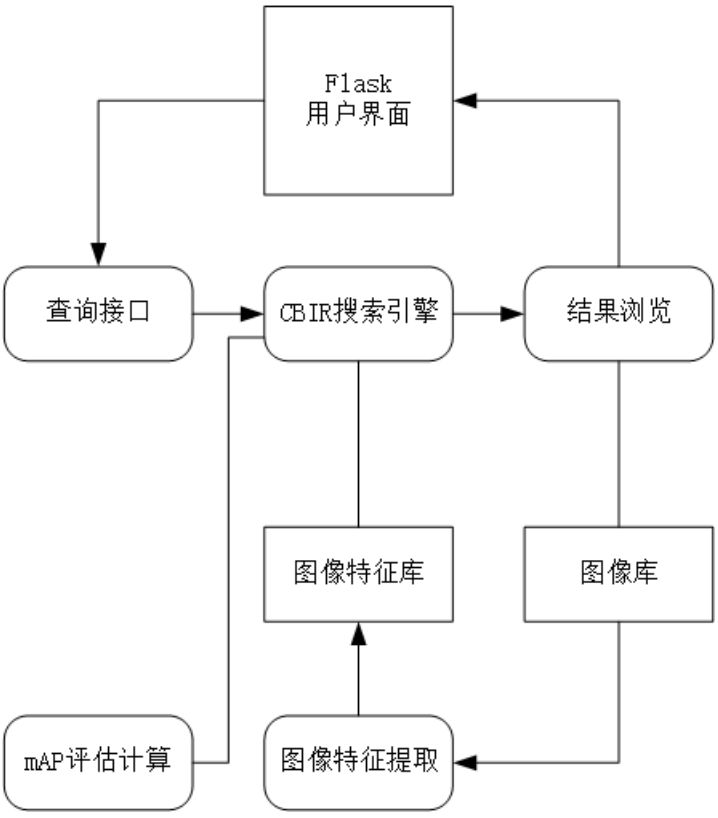
1.2.3 训练参数（已经训练过则可跳过）：python compute_features.py。

1.2.4 运行检索界面：python app.py，网页默认运行在 http://127.0.0.1:5000/

1.2.5 评估 mAP：python compute_map.py，结果输出在命令行。

2 系统设计

2.1 系统框图



2.2 组成模块（树状图）

- ImageRetrievalSystem ：代码文件夹。
- | app.py ：Flask 启动入口和路由等。
- | compute_features.py ：构建特征数据库（核心算法）。
- | compute_map.py ：完成评估任务的模块。
- | features.pkl ：特征数据库文件。
- | requirements.txt ：Python 环境依赖。

- │ searcher.py : 内含完成查询/检索任务的函数。
- │
- ├──static : 网站静态文件夹。
- │ │ index.css : index 页面使用的 CSS 样式文件。
- │ │ index.js : index 页面使用的 JavaScript 样式文件。
- │ │ preview.png : index 页面使用的预览图片。
- │ │
- │ ├──gt_files : 符合我的评估函数格式的 Ground Truth。
- │ │
- │ └──images :图像数据库文件夹,需要将 5K 张图像置入。
- └──templates : 网站模板文件夹
- index.html : HTML 网页文件。

2.3 任务描述

2.3.1 特征数据库构建任务：输入图像库，根据核心算法，进行长时间的运算，获得图像特征库，每次更换数据库时需要重新构建特征数据库。

2.3.2 前端任务：提供查询接口，允许用户在 Oxford5K 数据集中任选一张图片（如将自己的图像置于数据集目录则也可选择），请求后端提供查询结果，即和选中图像包含相同实例的 10 张图像，并展示 10 张图像及其名称和相似度评分。

2.3.3 后端任务：根据要求，计算指定图像的特征（没有直接从数据查询是因为允许查询图像是未知图像），并按评估排名的顺序从数据库返回包含相同实例的前 10 张图像机器名称和相似度评分给前端。

2.3.4 评估任务：根据指定的查询要求和 ground truth 对 Good、Ok、Junk 的判断，计算出相应的 Top10 mAP。（更改源代码也可以计算 Top5K mAP）

3 核心算法设计

3.1 算法原理

3.1.1 SIFT 和 SURF：尺度不变特征变换和加速健壮特征，使用时需要注意的是 SURF 是 64 维的，SIFT 是 128 维的，由于在第二次作业中使用过，在此不再赘述。

3.1.2 K-means：基于划分的聚类方法，在第二次作业中也使用过，在此不再赘述。

3.1.3 Tf-idf 加权和 L2： 我实现的搜索引擎使用的是信息检索的向量空间模型，就是将查询和语料库中的每个文档表示为视觉单词出现的稀疏向量，并通过 L2 距离计算查询向量与每个文档向量之间的相似度来进行搜索。还使用了标准的 tf-idf 加权方案来降低常见单词的权重，增强了鲁棒性，但是也导致了评估得分降低。

3.2 算法流程

3.2.1 构建特征数据库

3.2.1.1 提取 SURF 特征：提取 5K 张训练样本的 SURF 特征（速度和健壮性比 SIFT 更好）。

3.2.1.2 K-means 聚类：使用 K-means 方法对特征描述符进行聚类，结果就是 BOW 的词汇。

3.2.1.3 最后，使用之前计算的特征描述符和词汇计算出特征直方图，再使用 tf-idf 算法计算词频进行矢量化，最后对特征直方图进行 L2 归一化，将上述计算结果保存至 pkl 文件。

3.2.2 检索：从 pkl 文件加载参数，对目标图像提取 SURF 特征描述符后使用 Tf-Idf 算法进行矢量化，再将得到的图像特征和特征数据库的特征直方图进行点乘，即可获得和数据库中所有图像的相似度。

4 系统实现

4.1 算法实现（Compute_features.py）

4.1.1 SURF 特征提取：使用了 cv2.xfeatures2d.SURF_create 获得了特征描述符提取器，在提取特征前，使用 cv2.cvtColor 和 cv2.COLOR_BGR2GRAY 对图像进行了灰度化处理，移除色彩的干扰。

4.1.2 K-means 聚类：使用了 OpenCV 对 BOW 问题专门设计的 cv2.BOWKMeansTrainer，可以直接把特征描述符喂给 Trainer，很方便。

4.1.3 计算特征直方图：使用了 scipy.cluster.vq 进行辅助运算，这一步没有什么技术含量。

4.1.4 Tf-Idf 加权：核心思想是把“查询串 q 和文档 d 的匹配度问题”转化为“查询串 q 来自于文档 d 的条件概率问题”，搜索或评估时用它来判断两份文件之间的相似性。

4.1.5 L2 归一化：用之前的特征直方图和 idf 相乘，再对其进行 L2 归一化，防止过拟合。

4.2 界面实现

4.2.1 因为前端界面实现应该不是本课的重点，因此本次实验前端使用的是 GitHub 上的 Flask 界面实现，我将该实现的后端修改成了我自己实现的算法，效果很好。出于对前端原作者的尊重，再加上我很喜欢原作者对前端界面的设计，因此我在这里没有过多修改，只修改了小部分 HTML、CSS、JavaScript 代码，后端连接上了我的 Python 算法实现部分。

4.2.2 详细请见 2.2 组成模块（树状图）、5.2 系统测试和展示视频。

4.2.3 界面参考：https://github.com/kudeh/image-search-engine

5 实验

5.1 核心算法评价

5.1.1 实验数据集：Oxford5K

5.1.2 实验参数设置

5.1.2.1 K-Means K：200（受限于算力，CPU100%运行仍花费了 4 个小时），远小于论文最小 K 值。

5.1.3 算法性能展示

5.1.3.1 为了评估性能，原文使用平均精度（AP）度量作为查询的精确度调用曲线下的面积进行计算。精度定义为检索到的正图像/检索到的总数。召回定义为语料库中检索到的正图像数量/正图像总数。理想的精度调用曲线在所有调用级别上的精度均为 1，这对应于平均精度为 1。

5.1.3.2 根据我的理解，原文为每一个地标指定了 5 个查询，每次查询取前 10 的结果计算 AP，最后将总共 55 次查询的 AP 计算平均值以获得 mAP 得分，以评估整体表现。由于数据集自带的 ground truth 不是很好用，且提供的计算程序不能直接计算全局的 mAP，所以我修改了 gt_files 文件，使其适用于我的 compute_map.py。

5.1.3.3 Top10 mAP：0.648

```
Python Console
Top 10 mean Average Precision:  0.6475834764882383

>>>
```

5.1.3.4 Top5K mAP：0.165

```
Python Console
Mean Average Precision:  0.16472302119948667

>>>
```

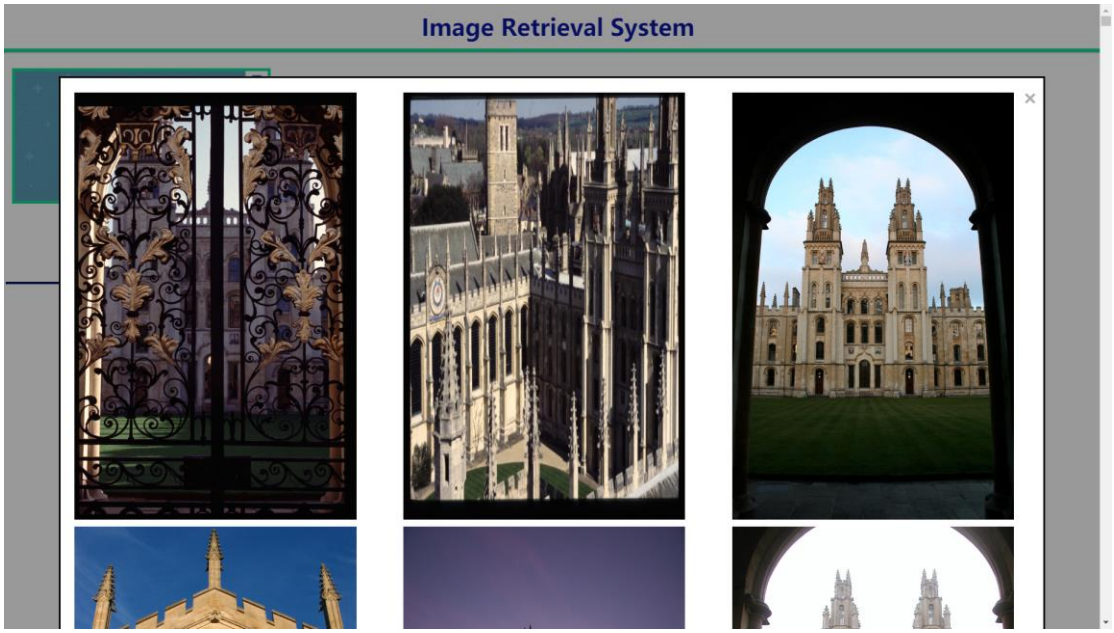
5.1.4 问题分析

5.1.4.1 问题 1：算力不足，导致在 K-Means 步骤时不能使用论文中使用的较大的参数，比如 1000 或以上数量级，这点应该是主要问题。

5.1.4.2 问题 2：没有使用论文中建议使用的 AKM 和 HKM，但是按照原文的说法，其实对性能的影响很有限。

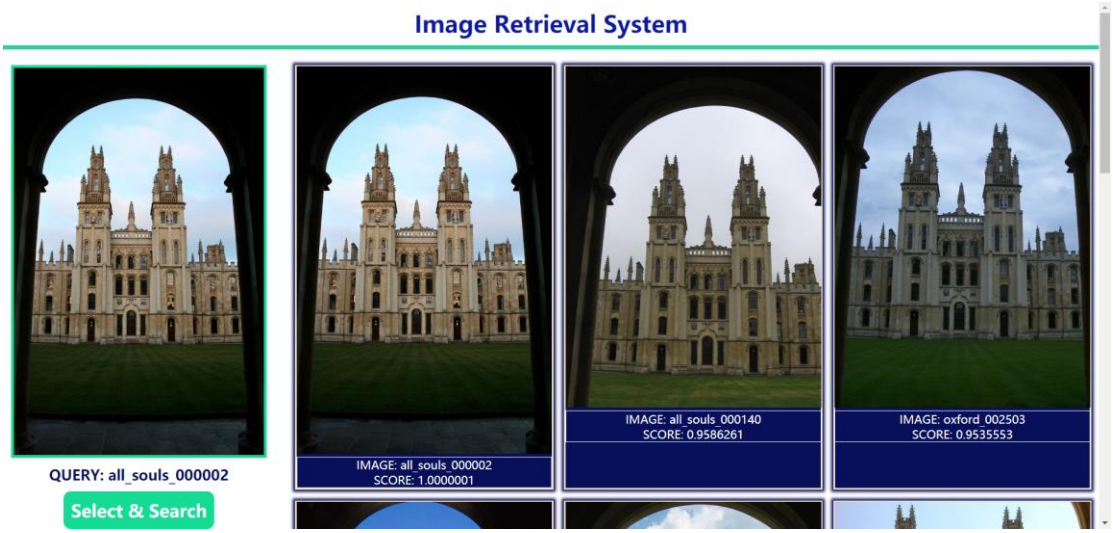
5.2 系统测试

5.2.1 界面功能介绍：初始界面如下图，点击左侧按钮会弹出一个模态框，如下下图所示，在模态框里从下至下有全部的 5K 张图像，可任选一张来进行图像检索，如果想使用自己拍的照片，可以将它置于图像数据库的目录下，这样就能在模态框中找到它；在模态框中选中一张图像后，系统会自动开始检索任务，进入下一阶段。



5.2.2 系统工作过程：本阶段持续时间只有几秒，此时后端在计算选中图像的特征，并将其与图像特征库的图像特征直方图矩阵进行点乘，得到排名得分，并在进行后处理后将结果传递给前端，即可得到与该图像内容最像的 10 张图像及其名称和相似度评分。

5.2.3 结果展示：如图所示，结果会以原图像比例展示，不会拉伸收缩，因此部分图像的下方有蓝色空隙。在内容上，显示了与选中图像内容最像的 10 张图像及其名称和相似度评分。



6 结论

6.1 系统的优势

6.1.1 CPU 还可以的笔记本运满载行 4 小时即可生成特征数据库，训练时间在可以忍受的临界线内；

6.1.2 使用了 SURF，它的区分性不逊色于 SIFT，但是在速度和鲁棒性较 SIFT 有明显优势，在大规模运算时相比 SIFT 具有压倒性的优势，实际测试也是如此；

6.1.3 图像特征数据库文件很小，只有 3.5MB，而且检索速度极快，只需要一秒左右。

6.2 系统的缺点

6.2.1 因为算力受限，又不想减小训练用的数据集，所以 K-means 的参数 K 只有 200，难以涵盖 5K 张图像的词汇特征，再加上很多质量较差的图像没有在训练时被特殊对待，因此尽管正常使用时效果很好，但是在评估环节可以看出总体性能较差；

6.2.2 但是我认为这也能说明论文的局限性，原论文一直在讨论在大型数据集进行图像检索的提升，但是却只字不提各种实验参数和方法所使用的算力或消耗的时间，这其实对于大型系统影响还是很大的，这一点可能是提升或研究方向之一，可能的提升方向是使用时间复杂度更低的算法替换已有算法；

6.2.3 在性能方面，本系统的实现原理于近 15 年前已经提出，已经有些落后于时代了，可能的提升方法是使用深度学习方法来代替部分传统的计算过程；

6.2.4 此外，没实现论文中的 AKM 和 HKM 来代替 K-means，如果实现可能可以小幅提升性能。