

图像搜索引擎 Doodle 实验报告

2017011474 计73 郑林楷

2017011387 计76 徐天行

图像搜索引擎 Doodle 实验报告

问题描述

实现模块

网站前端

网站后端

图像检索系统

条件筛选器

尺寸筛选

颜色筛选

关键功能

基于内容的以图搜图

相关图片推荐

主色调筛选与尺寸筛选

测试结果

基于内容的以图搜图

相关图片推荐功能

主色调筛选

样例分析

基于内容的以图搜图

相关图片推荐

主色调筛选

开源资料

开源代码

小组分工

项目代码

问题描述

题目：按照查询词返回相关图片并进行排序（利用图片原网页内容或者图片视觉信息等），并对返回的图片进行一定的布局，扩展功能包括：按照图片的尺寸、颜色进行筛选展示搜索结果。可用数据集：[INRIA Holidays dataset](#)，为世界名胜图片集合，共 1491 张图片，包括 500 个查询以及其对应的 991 张相关图片，共 3G。

分析：INRIA Holidays dataset 是图片检索任务的经典数据集。基于内容的图像检索任务是根据图像、图像的内容语义以及上下文联系进行查找，以图像语义特征为线索从图像数据库中检出具有相似特性的其它图像。而由于 INRIA Holidays dataset 缺乏文本信息，只有图像信息，因此我们的问题描述如下。

问题描述：给定一张图片，要求从 INRIA Holidays dataset 中找出内容上与给定图片相关的所有图片，并根据相关性进行排序。可以按照图片的尺寸、颜色筛选展示的搜索结果，但是展示顺序只与内容相关性有关。

实现模块

我们的 Doodle 以图搜图图像检索网站可以简单的分成以下三个部分：

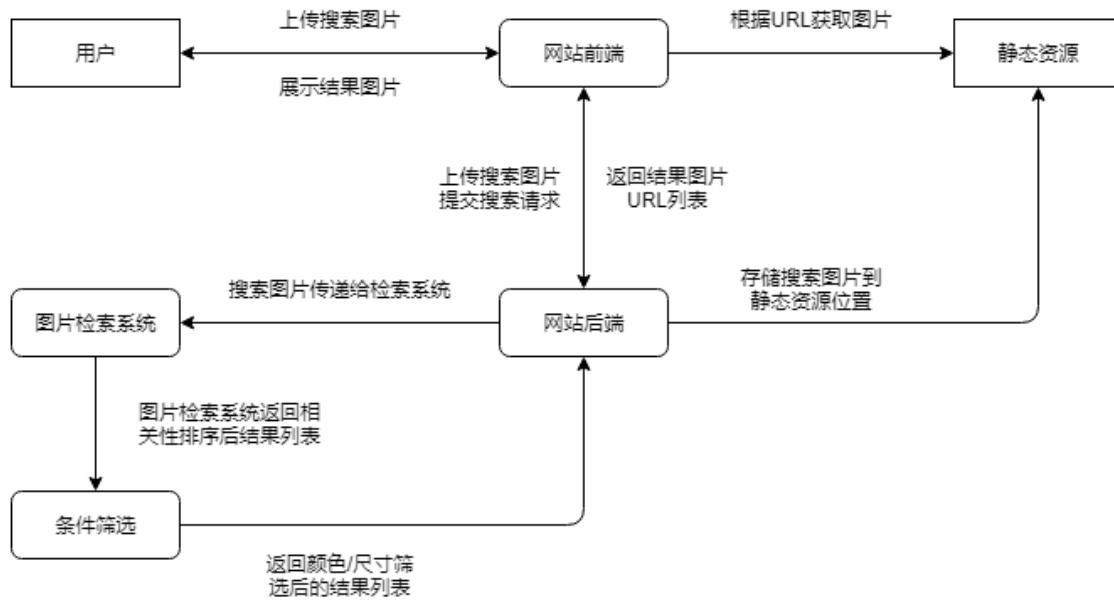
- 展示和交互用的网站

- 基于内容的图像检索系统，对图片集进行内容相关性排序
- 条件筛选器，进行尺寸和颜色的筛选

其中网站部分包含以下两个部分：

- 使用 Nuxt.js + Vuetify.js 搭建的网站前端
- 使用 Flask 搭建的网站后端

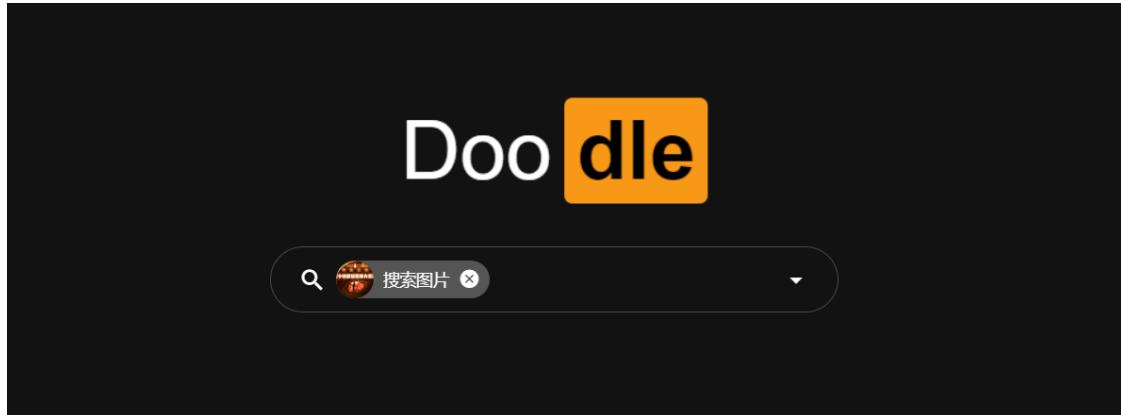
各模块之间的调用关系如下图所示：



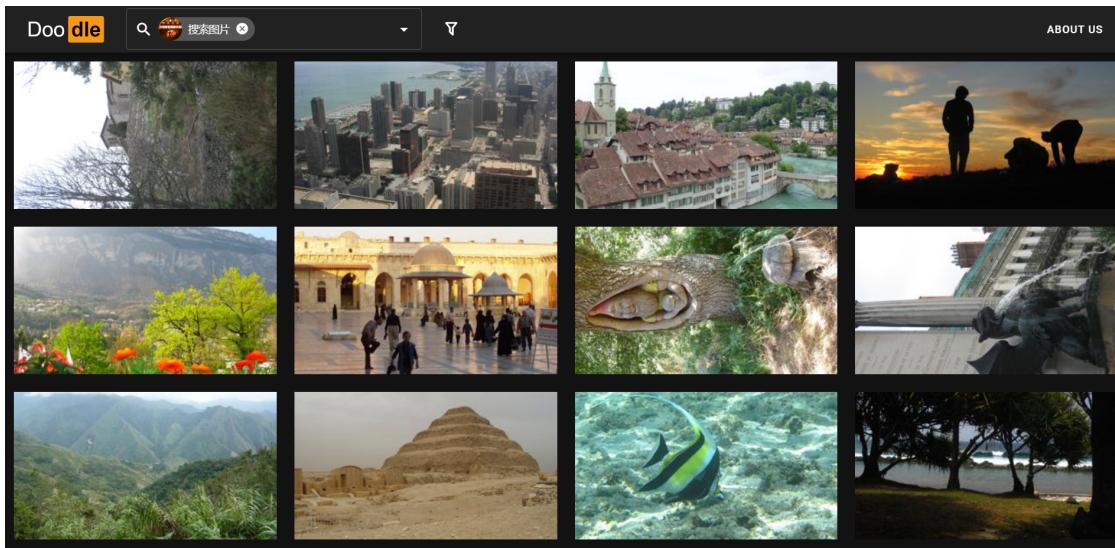
网站前端

网站前端采用 Nuxt.js 实现，使用 Vuetify.js 作为前端的UI显示，界面美观。在布局方面，我们参考了 Google 的图片搜索界面布局，尽量贴合用户的使用习惯，减少用户的上手难度，设计图如下所示：

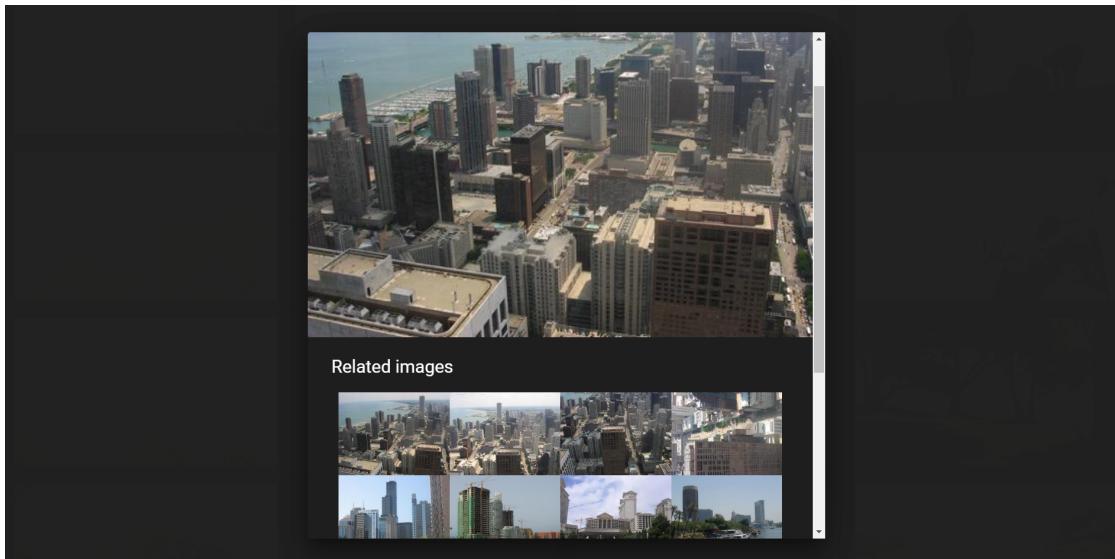
- Doodle 网站主页：用户可以在此上传图片并进行搜索



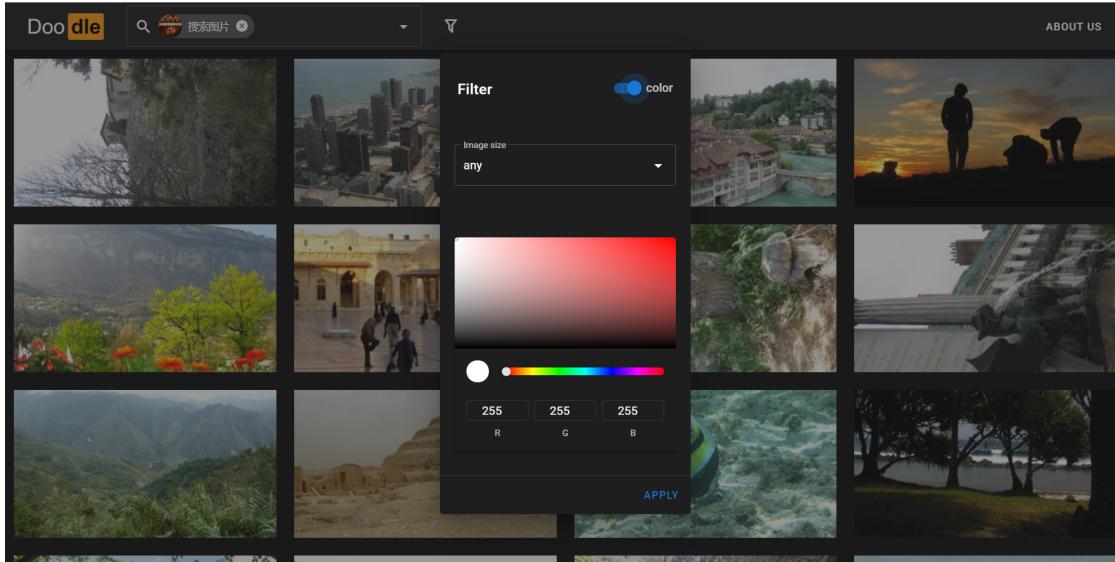
- Doodle 搜索结果界面：图片的搜索结果在此显示，Doodle 根据图片内容关联程度进行排序，将相关性高的图片放到前面



- 图片详情弹窗：点击搜索结果中的缩略图，用户可以观看图片的大图展示，弹窗下半部分显示 Doodle 推荐的与该图片相关联的其他图片



- 条件筛选器：点击结果界面顶部栏的筛选图标，可以添加筛选条件



网站后端

网站后端基于 Flask 实现，主要负责前端和图像库以及图像检索系统的交互。我们在网站后端连接了图像检索系统，并提供了 RestAPI 接口供前端调用：

- `/api/search`：基于查询图和筛选条件下查询图像库内的相关图像
- `/api/relate`：查询图像库内某张图像的相关图像

- `/api/upload` : 上传图像作为查询图
- `/img/<imgID>` 和 `/upload/<imgID>` : 返回图像数据

由于图像库中的每张图片大小都在 1MB 以上，我们发现搜索结果页中一次性加载 20 张原图所需要的时间过于久。为了提升用户体验，我们给 `/img/<imgID>` 附带了动态尺寸参数 `s=AyB`，其中 A 和 B 对应图像的长和宽。此参数会限定返回图像的尺寸，网站后端会动态调整图像的尺寸并将调整后的图像数据返回。基于此参数，我们实现了缩略图显示和详情图显示的功能，搜索结果页的加载速度得到了明显的提升。

各 RestAPI 的具体用法见 `API.md`。

图像检索系统

经过仔细的研究和分析，发现该图片检索任务具有以下特点：

- 使用的图片集为 INRIA Holidays dataset，数据集规模不大，图片个数只有 1491，但是图片的分辨率较高，很多图片大小超过3MB。
- 尽管 Nuxt.js 支持分布式后端服务器部署，但是客观条件所限我们没有足够多的服务器来进行实验，而高并发性能的提升可以通过更好和更多的分布式后端服务器来实现，在客观条件的限制下考察并发相关指标没有意义。
- 在搭建好网站后我们进行测试，发现客户端从发起请求到获得响应的时间瓶颈在于网络传输，网站需要同时显示若干张图片。尽管后端对图片进行了压缩处理，但是数据传输时长仍然堪忧。而且，该传输时长可以通过购买带宽，将静态资源部署到 CDN 上进行优化，在客观条件限制下考察延时也没有太大意义。

因此，我们认为我们开发的图片搜索引擎应当是一款对并发性和响应延时相对宽容的，以搜索引擎效果为优化指标的，允许对图片集合进行复杂预处理操作的图片搜索引擎。我们最终决定采用深度学习模型完成该图片检索任务。

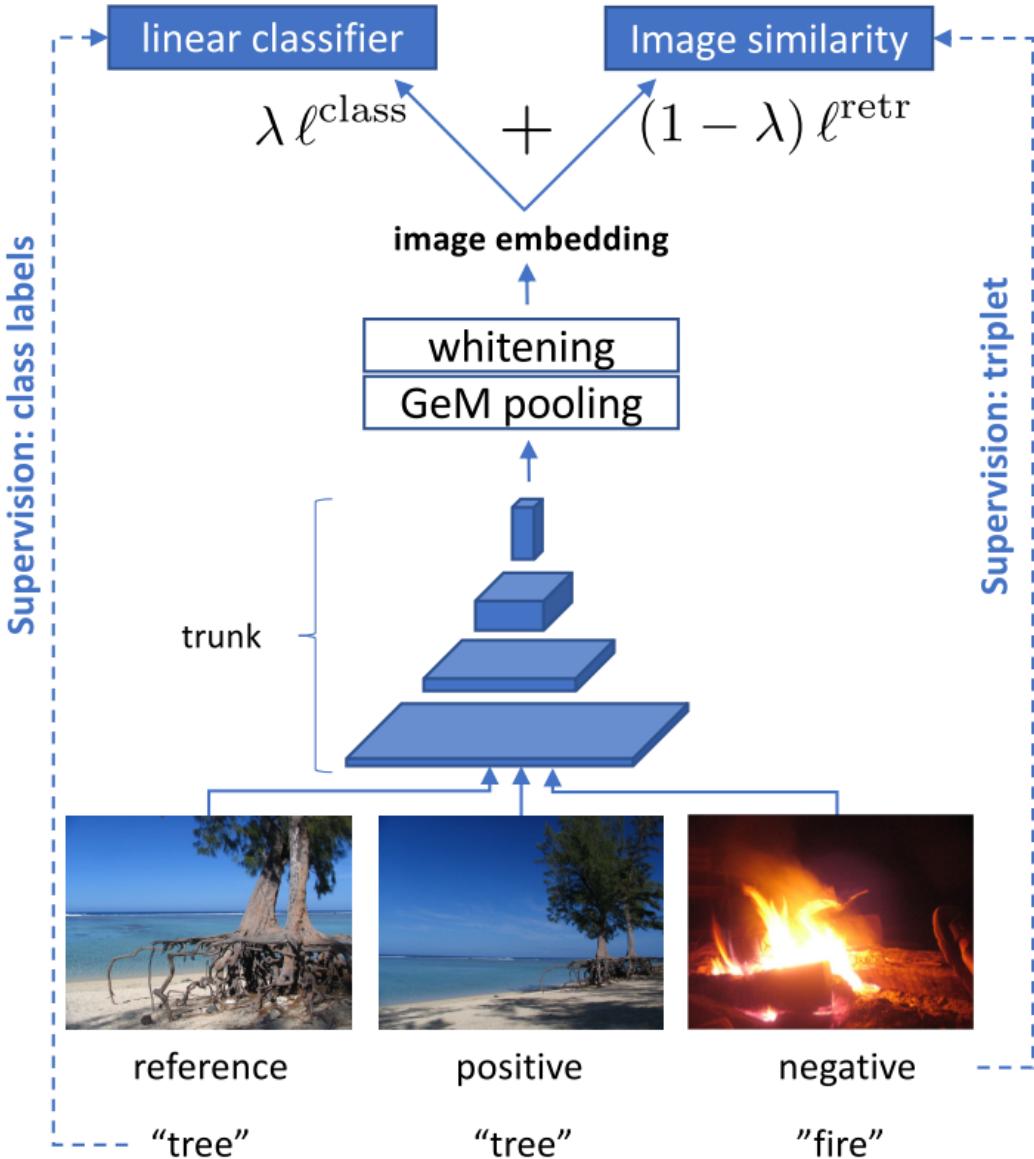
图片检索任务的通用架构如下所示：

- 图片特征提取：使用深度学习模型提取图片的语义特征
- 特征降维：使用主成分分析，白化等手段进行特征降维，节约存储空间，加速计算时间
- 特征比对：使用合理的距离函数衡量特征之间的距离，以此作为图片关联度

由于我们对响应延时要求较低，以效果作为优化指标，因此我们最终没有进行特征降维。图片检索系统对于搜索图片的处理流程如下：

- 使用特征提取模型对传入的搜索图片进行特征提取
- 计算搜索图片的特征与图片集中每张图片预处理出的特征向量之间的距离
- 根据计算的距离对图片集中的图片进行排序，返回排序好的图片列表

经过多篇 CBIR 相关论文的研究，我们选用 Facebook 在 2019 年提出的 MultiGrain 架构进行图片召回，MultiGrain 架构如下所示：



MultiGrain 架构支持使用任意一种图片识别模型作为特征提取模型，可以根据网站后端部署的服务器性能差异进行调整，支持 Mobile Net v2 到 Dense Net 121 不同参数规模和计算量要求的模型。而 GeM 池化层的使用支持模型将任意尺寸的输入图片转化为 2048 维的向量，避免在图片预处理过程中，图片剪裁影响模型表现。最后，MultiGrain 统一了图片识别任务和图片检索任务，在训练模型的过程中便于复用图片识别模型的参数。

我们使用 Google 在 2018 年提出的 PNASNet-5 作为基础模型，并在 500*500 大小的 ImageNet 上进行 Finetune，根据 MultiGrain 论文中的结论，分辨率更高的数据集上进行 finetune 有助于改善模型的表现。

距离函数的选取需要与模型训练时 Triplet Loss 中损失函数的定义一致，我们使用的是 MultiGrain 的预训练模型的参数，因此距离函数使用 L2 正则化处理后的特征向量的 Euclidean 距离。

条件筛选器

条件筛选包括尺寸筛选和颜色筛选。

尺寸筛选

尺寸筛选方面，我们根据图片原大小的长宽分为 Small, Medium, Large 三类进行筛选。设图片长为 H ，宽为 W ，则筛选条件为：

- 当 $\max(H, W) \in [0, 2000]$ 时归类至 Small

- 当 $\max(H, W) \in (2000, 3000]$ 时归类至 Medium
- 当 $\max(H, W) \in (3000, \infty]$ 时归类至 Large

颜色筛选

颜色筛选方面，我们对图像库中的图片做预处理，提取出每张图片的主要颜色，再根据筛选色对每张图片进行评分，最后筛选掉阈值以下的图片并对结果排序后返回。

首先是预处理部分。我们使用了 2010 年一篇名叫《基于 OTSU 分割与 K-均值聚类的 MPEG-7 主颜色提取算法》的论文内所提及的「K-均值聚类算法」主颜色提取算法对图像库内的图片进行预处理。算法主要步骤如下：

- 将图像的各像素的颜色值从 RGB 颜色空间转换到 HSV 颜色空间
- 确定图像的聚类数目 k 和初始聚类中心 MCC
- 计算每个像素点与聚类中心 MCC 在 HSV 颜色空间的 xyz 三维欧几里得距离，并将其归到与该像素点最近的聚类中
- 重新计算每个聚类的聚类中心，生成新的 MCC
- 若 MCC 变化幅度在阈值内，则不再进行递归，否则返回第三步
- 统计各聚类的像素数，得到图像的若干主颜色和其对应的比例

对于算法第二步，我们采用「H 直方图峰值筛选法」确定初始聚类中心及其数目。算法主要步骤如下：

- 对各像素点颜色中的 H 分量做直方图，组数 255， $h(i)$ 表示第 i 组的频数
- 找初始峰值：初始峰值的集合 $P_0 = \{i | h(i-1) \leq h(i) \wedge h(i) \geq h(i+1), 1 \leq i \leq 255\}$ ，注意当 $i = 255$ 时 $h(i+1) = h(0)$
- 找明显峰值：明显峰值的集合 $P_1 = \{i | h(i-1) \leq h(i) \wedge h(i) \geq h(i+1), i \in P_0\}$
- 去除小峰值：设 t 为像素点总数，则将 $h(i) < 0.5\% \times t$ 的小峰值从 P_1 中去除
- 去除相邻的峰值：若有两个峰值组号的距离在 30 以内，则认为这两个峰值所代表的区域颜色相似，只保留 $h(i)$ 较大的峰值

第一步只考虑 H 分量是因为在 HSV 颜色空间中，人眼对色调 H 最为敏感，故只对 H 分量的直方图进行峰值筛选处理。

进行颜色筛选时，我们根据筛选色 F 对图像库的每张图片计算出各自的得分，最后保留得分在 1.5 以下的作为筛选结果。得分计算公式为：

$$\text{Score} = \sum G(F, C_i) \times P_i$$

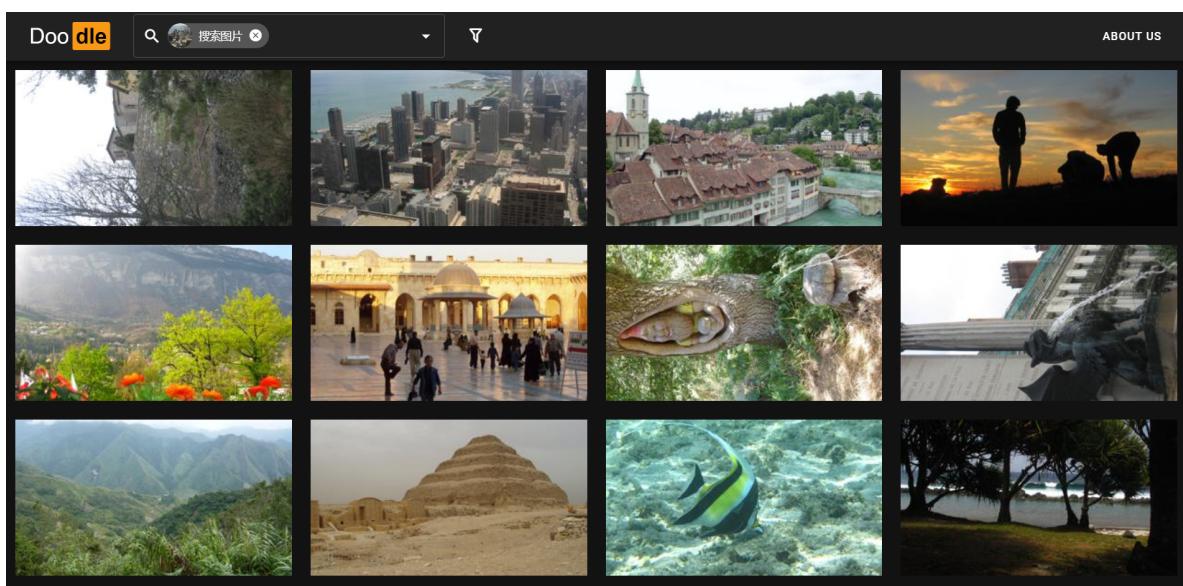
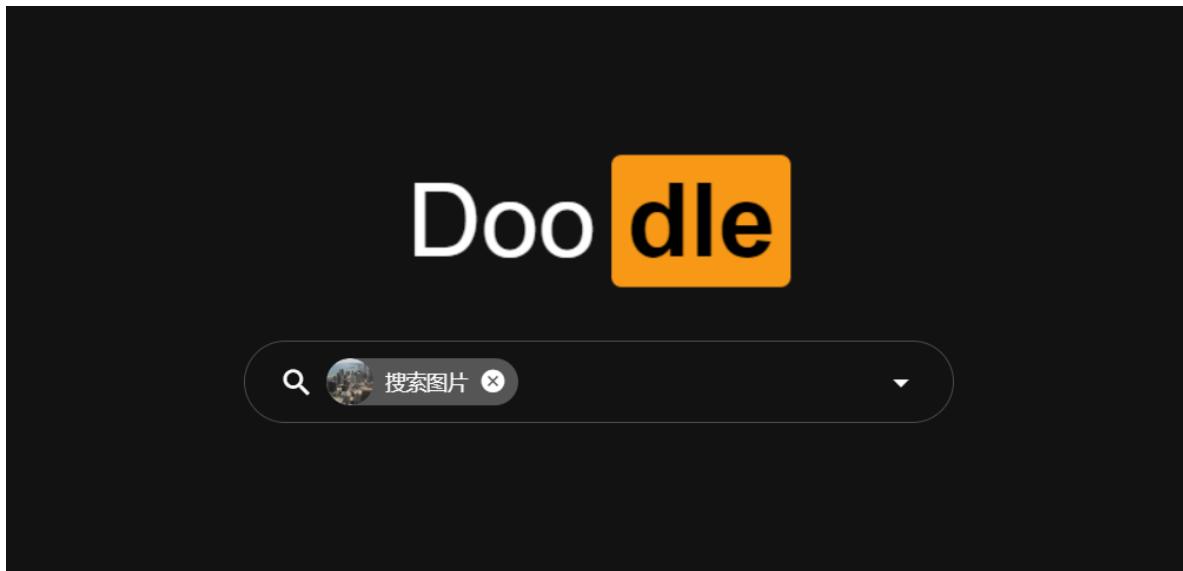
$$G(A, B) = \begin{cases} \text{dis}(A, B) & (\text{dis}(A, B) \leq 0.3) \\ 2 & (\text{dis}(A, B) > 0.3) \end{cases}$$

其中 C_i 为该图的第 i 个主颜色， P_i 为其在图中所占比例，dis 函数为两颜色在 HSV 颜色空间内的 xyz 三维欧几里得距离。

关键功能

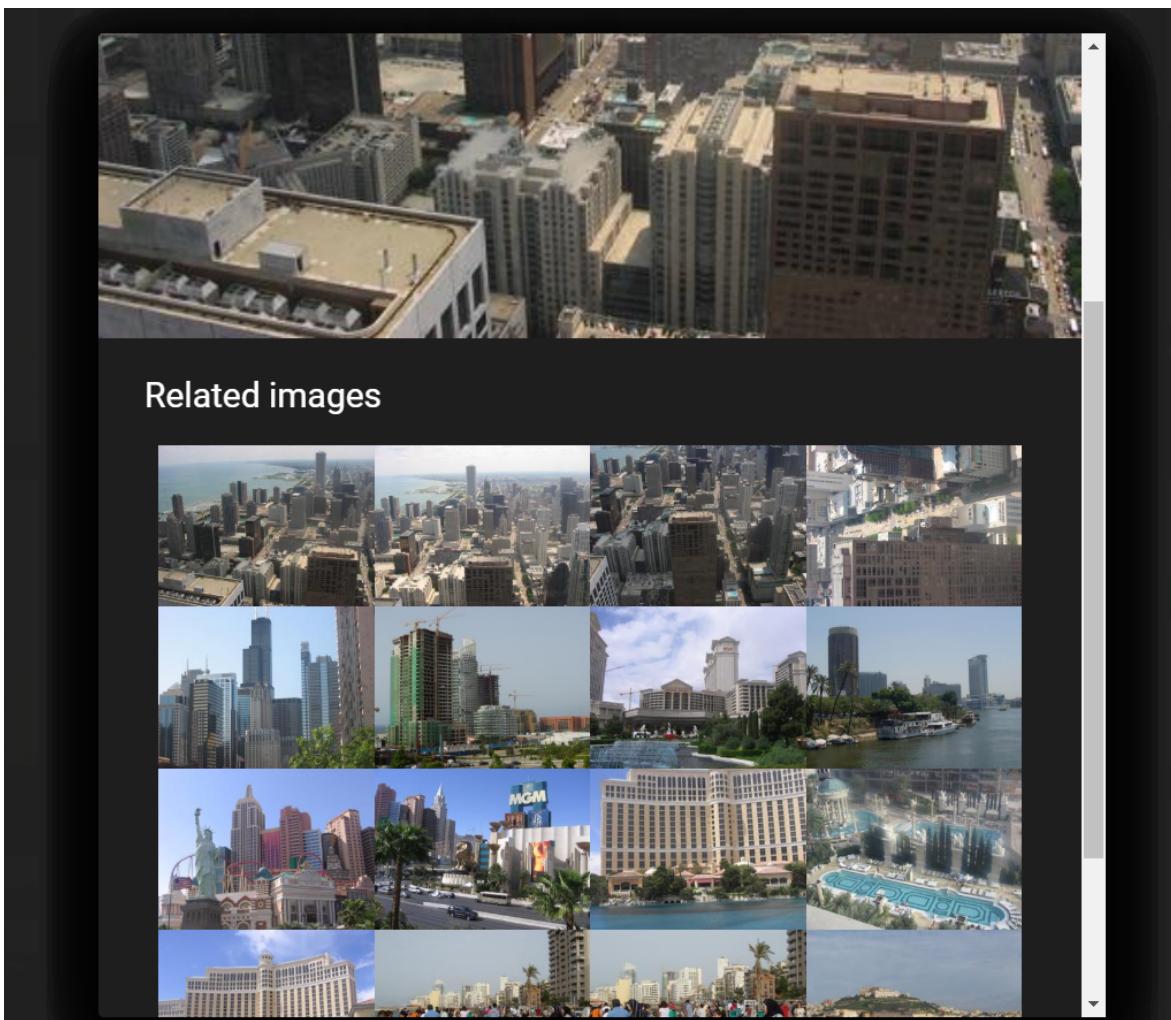
基于内容的以图搜图

基于内容的以图搜图支持 Doodle 引擎用户上传自定义图片，以此作为关键词进行图片检索，返回若干图片，图片按照相关性进行排序。



相关图片推荐

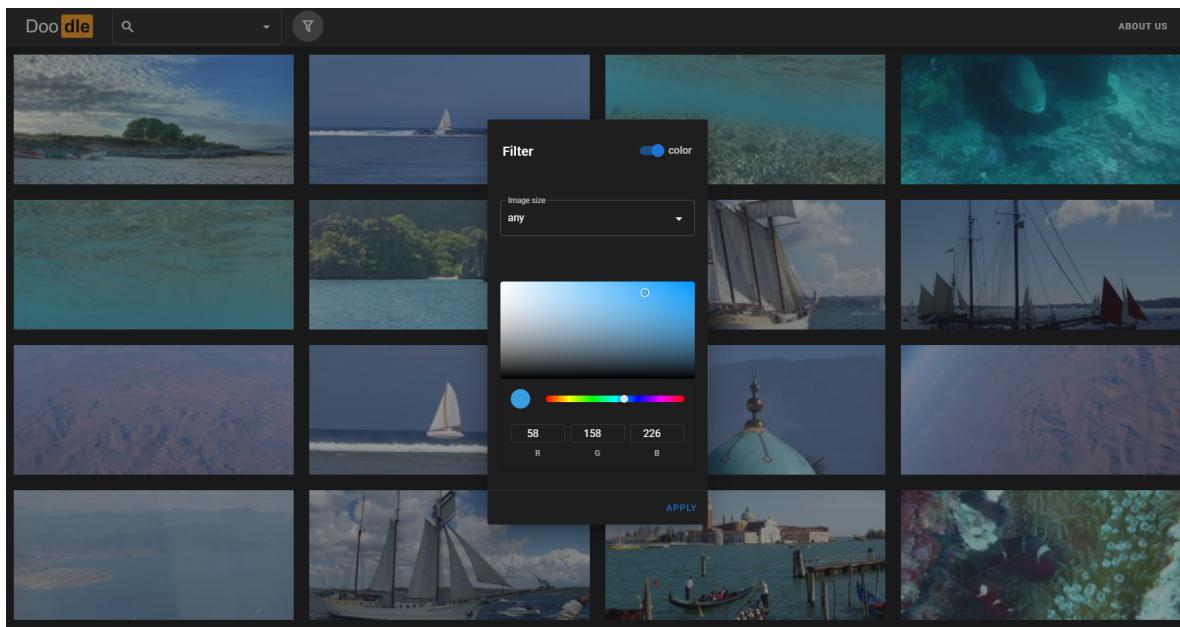
相关图片推荐功能支持用户点击搜索结果列表中的图片，在详情页面下展示与该图片内容相关的其他图片。



主色调筛选与尺寸筛选

主色调筛选与尺寸筛选功能支持用户指定颜色和尺寸，对搜索结果中显示的图片进行筛选，以红色和蓝色为例：

The screenshot shows the Doodie app's interface for image filtering. On the left, there are several image thumbnails. In the center, a 'Filter' panel is open. It includes a 'color' toggle switch, a dropdown menu set to 'any', and a color picker with sliders for Red (R: 204), Green (G: 54), and Blue (B: 54). At the bottom of the filter panel is an 'APPLY' button. The thumbnails on the right side of the screen are predominantly red or blue, indicating they have been selected by the current filter settings.



测试结果

Doodle 搜索引擎包含三部分功能，因此针对三部分的功能分别进行性能的评价。搜索引擎性能评价的关注对象主要包含两部分：

- 搜索引擎系统运行效率：这部分性能评价指标可以通过使用更好的服务器，将静态资源部署到 CDN 上，以及使用 GPU 和分布式系统进行加速来获得更好的结果。目前 Doodle 的时间瓶颈在于网络传输，因此本次实验的性能评价不包含这一部分
- 搜索引擎系统运行效果：Doodle 性能评价的核心部分

基于内容的以图搜图

我们随机选取 10 张图片测试以图搜图，所选图片全部来自网络，不存在于图片集中，因为 Doodle 每页显示 20 张搜索结果，所以我们主要考虑结果列表中的前 20 张图片，关注指标 AvgPrecision@20。使用 01 表示该图片与搜索图片是否相关。当用户不滚动时，可以看到第一页的前 12 张图片，因此我们还关注 Success@12 和 Precision@12。

内容	搜索图片	结果向量	AP@20	RR	S@12	P@12
山	test/1.jpg	1111 1111 1111 1111 1111	100%	1	1	100%
鱼	test/2.jpg	1111 1011 1111 1111 1111	93.4%	1	1	91.7%
高楼	test/3.jpg	1111 1111 1111 1001 0011	94.3%	1	1	100%
夕阳	test/4.jpg	1111 1111 1111 1111 0110	98.7%	1	1	100%
沙滩	test/5.jpg	1111 0000 0111 1111 1011	69.9%	1	1	58.3%
雪山	test/6.jpg	1111 1111 1011 1100 0001	90.6%	1	1	91.7%
小房子	test/7.jpg	1111 1111 1111 1111 1110	99.7%	1	1	100%
金字塔	test/8.jpg	1110 1100 1100 1001 0101	68.8%	1	1	58.3%
极地	test/9.jpg	1100 1001 0000 1000 1000	47.1%	1	1	33.3%
礁石	test/10.jpg	1111 1011 1111 1111 1101	92.9%	1	1	91.7%

针对不同的搜索内容，AvgPrecision@20 的平均值为 85.5%，Precision@12 的平均值为 82.5%。

相关图片推荐功能

在每张图片的详情页中，Doodle 将给出 20 张相关图片，因为是以缩略图的形式体现，不太需要翻页等操作，因此我们只以 Precision@20 作为衡量的指标，我们从图片库中任意抽取 10 张图片进行图片推荐功能，结果如下：

图片编号	精准推荐图片数	P@20
105104	19	95%
130302	20	100%
114002	17	85%
138303	10	50%
139400	16	80%
138102	12	60%
108103	11	55%
133300	20	100%
101400	19	95%
104201	18	90%

对于不同的图片，推荐的准确率 Precision@20 平均值为 81%。

主色调筛选

使用以下三种色调 #B71818 #18A4B7 #339A2A 进行筛选，得到的准确率结果如下：

颜色	色调正确图片数	准确率
#B71818	79/86	91.9%
#18A4B7	108/112	96.4%
#339A2A	50/50	100%

对于不同颜色的筛选，平均准确率为 96.1%，不过主色调相关与否主观感受占比比较大，因此颜色筛选的准确率定性为主，总的来说是一个较好的颜色筛选器。

样例分析

我们针对搜索引擎上表现不好的样例分别进行分析

基于内容的以图搜图

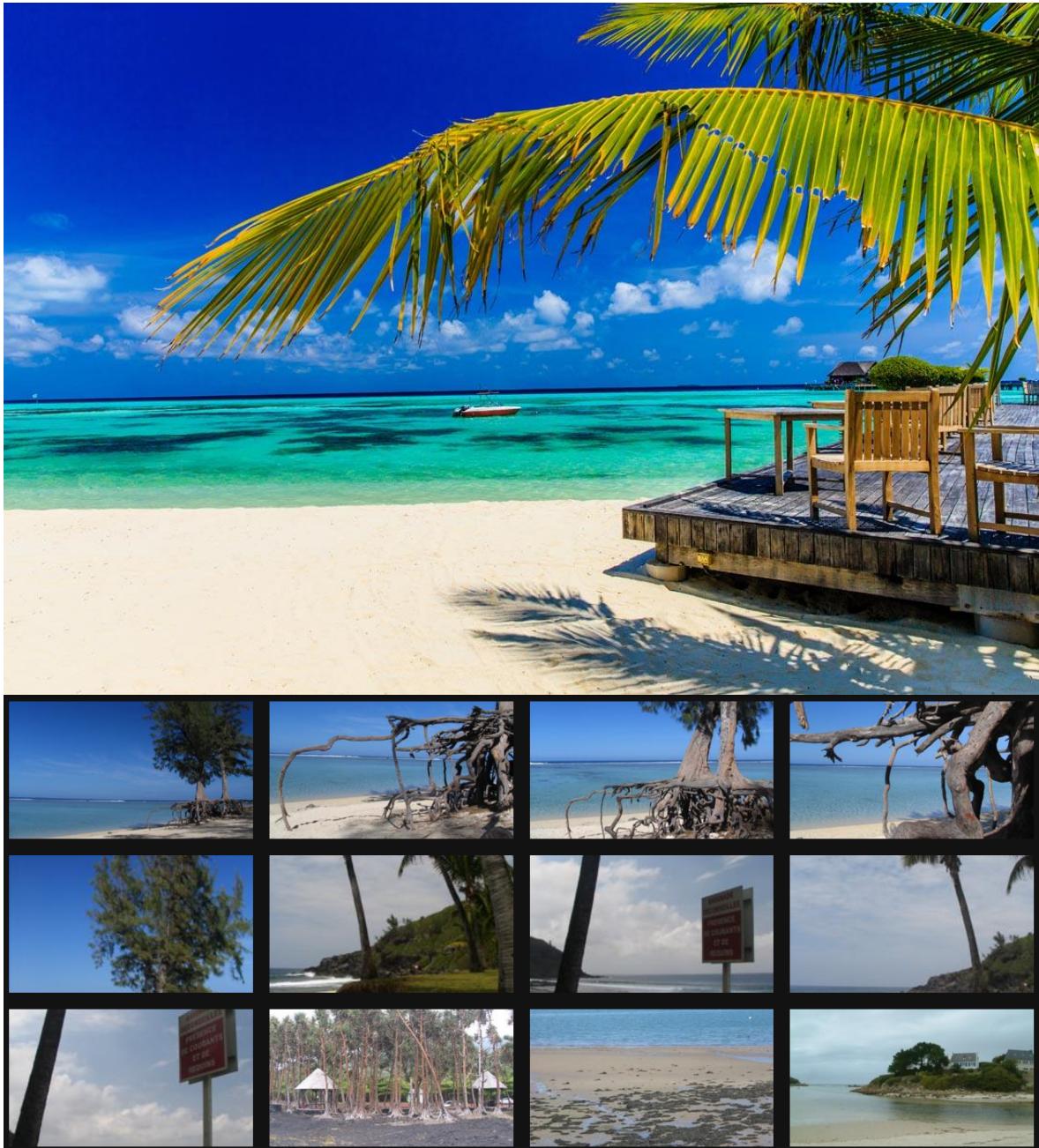
基于内容的以图搜图在以下关键词下显著差于其他结果，分别是：「金字塔」，「沙滩」，「极地」

以下是金字塔的搜索结果中前 12 张图片：



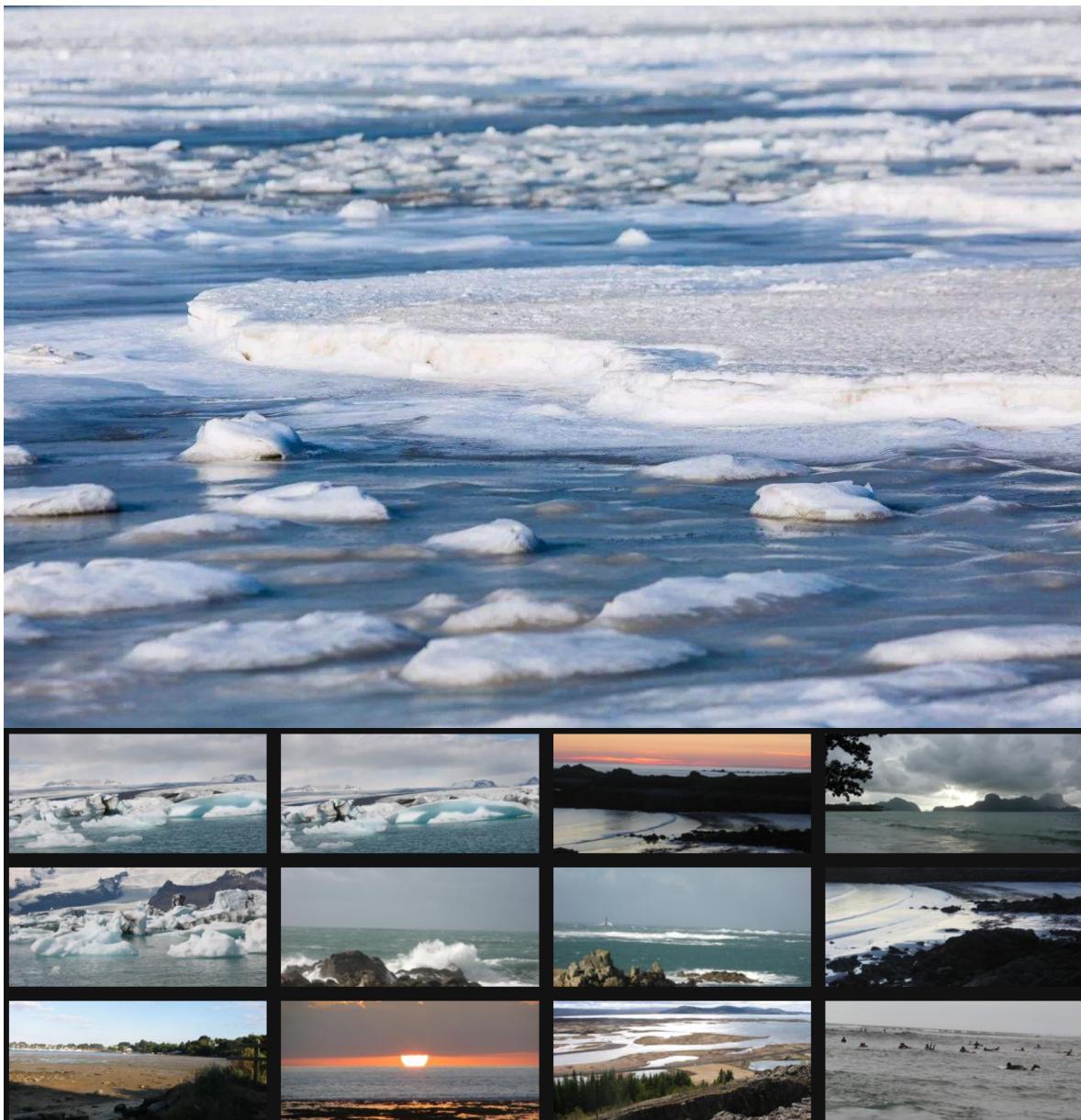
金字塔搜索结果中，第 4,7,8,11,12 张图片并不是与金字塔毫无关联，这几张图片与金字塔的色调接近，猜测拍摄对象为金字塔的局部，类似于第 5 张图片。但是由于没有金字塔的标志性三角外观，因此算作不相关。

以下是沙滩的搜索结果中前 12 张图片：



我们可以发现作为关键词的搜索图片中不止含有「沙滩」，还有「海洋」，「树木」，「天空」，因此在结果图片中我们也可以看到错误的图片中也是在这几类之中，而在统计时，只考虑图片中是否出现了「沙滩」这一元素，因此表现不佳。如果想要改善搜索引擎的表现，可以在上传关键图片时，对图片进行裁剪，以沙滩作为主体。

以下是极地的搜索结果中前 12 张图片：



极地图片表现不好的原因综合了以上两点，一方面搜索的关键图片语义信息为「海面上漂浮的冰」，包含「冰块」和「海」两个元素，而错误图片中都包含了「海」这一元素。另一方面，图像集中和极地相关的图像数量在 15 张左右，故搜索结果首页在显示完相关图片后，会显示各种非相关的图片，造成搜索结果变差。

相关图片推荐

以下是编号 138303 对应的图片和推荐结果：



Related images



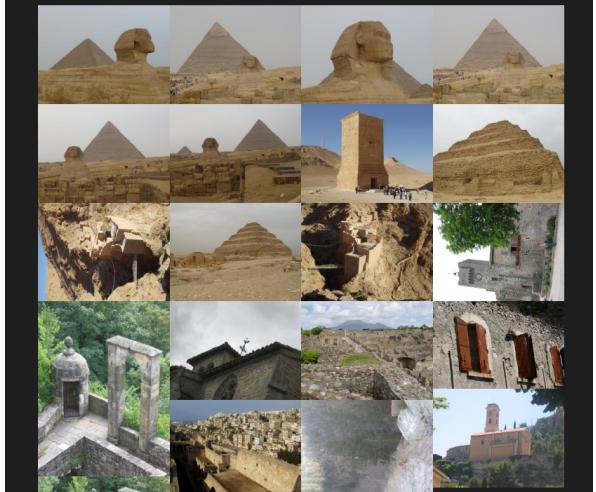
经过分析，我们发现上述测试中效果不好的样例的原因是图片集本身与水果摆盘相关的图片只有 10 张，因此对于固定推荐数量的 Doodle 来说，Precision@20 的理论上限就是 50%，与算法无关。

类似情况还有：

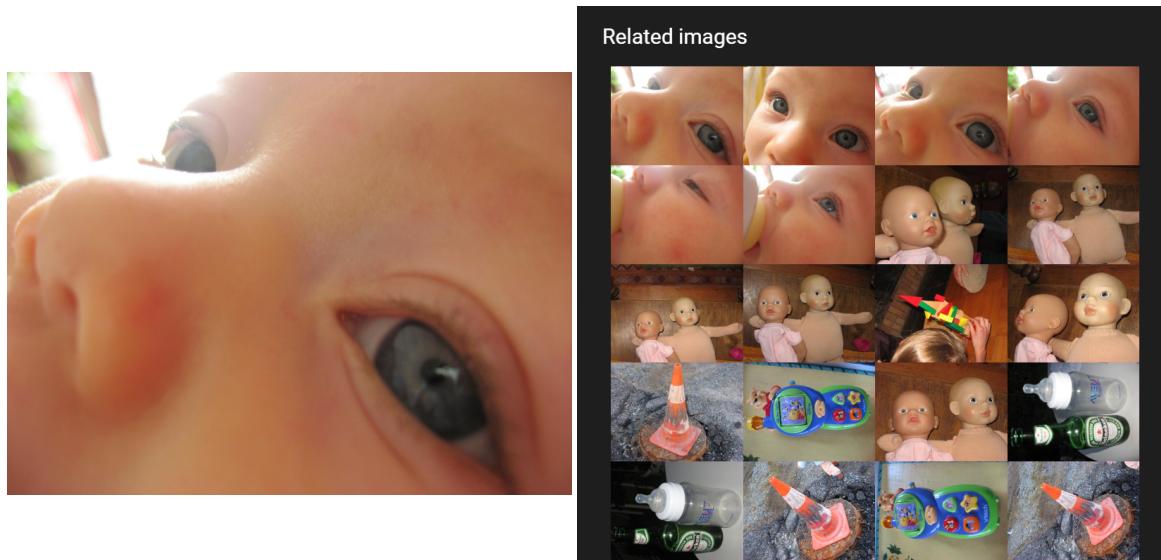
- 编号 108103



Related images

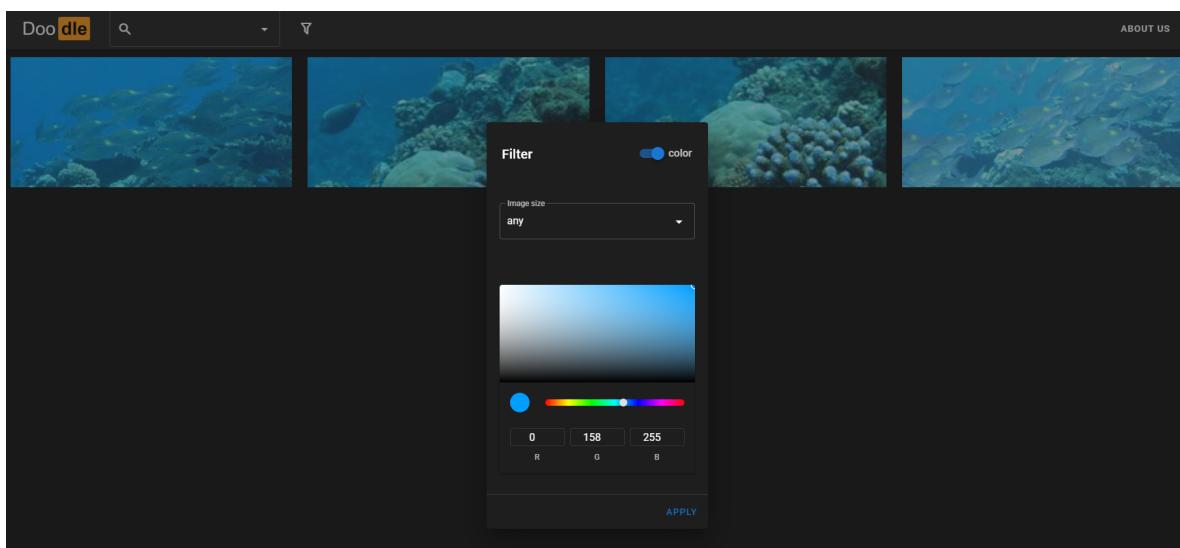


- 编号 138102



主色调筛选

我们在测试中发现，「主色调筛选」功能中若筛选色的饱和度和亮度都过于大，即在 HSV 颜色空间中 S 和 V 的值接近于 1，位置靠近 HSV 圆柱空间的边缘，则有些人眼看起来相近的颜色会不在其阈值空间邻域内，导致搜索结果中有些主色调相近的图片被遗漏，进而导致搜索结果数量少于预期。



开源资料

- [MultiGrain: a unified image embedding for classes and instances](#)
- [Learning Transferable Architectures for Scalable Image Recognition](#)
- [INRIA Holidays dataset](#)
- [Nuxt.js Document](#)
- [Flask Document](#)
- [Vuetify.js Document](#)
- [基于 OTSU 分割与 K-均值聚类的 MPEG-7 主颜色提取算法](#)

开源代码

- [MultiGrain](#)
- [Nuxt.js](#)
- [Vuetify.js](#)
- [Flask](#)

- [vue-image-crop-upload](#)

小组分工

- 徐天行：网站前端，图片检索系统，实验报告
- 郑林楷：网站后端，条件筛选器，实验报告

项目代码

项目代码位于 [GitHub 仓库](#)。