实验报告

依赖库：opencv、dlib、face\_recoginition、lucene（scipy、numpy之类的前置库就不列出了）

分工：

前端：王皓轩、徐迅

后端：卢锌原、王池

文字搜索：

1. 前期准备，主要是用爬虫爬取京东大量的商品网站，获取价格、热度、评论等信息。
2. 搜索及排序实现。主要是用Lucene来实验。用lucene自带的接口来实验建立索引。同时，由于Lucene自带的排序方式是相关度排序，在我们的实验中相关度的优先级应该是比较低的，因此用了Lucene提供 的sort类来实现自己的排序方式。主要是：按照价格，按照热度。同时还对网页进行了评分，与好评正相关，与差评、价格负相关的方式来获取一个网页的评分，然后按这个评分来进行排序。同时，亦可以对价格域进行限定，来进行搜索。
3. 在最后按评分来进行排序的过程中，我们发现有一部分价格极低的商品会大大影响搜索体验。这部分商品的热度往往不够，但凭借着低于同类商品平均价格很多倍的优势来获取极高的分数。因此，我们对索引进行了分层。只有当在优秀的商品中获取的数量不够时，我们才会对较差的商品索引进行补充搜索。

图像搜索：

1. 前期准备。爬取了口红官网来获取口红的色块与样品图。对色块用opencv进行处理，获得RGB，并将其同其他信息一同存储。
2. 唇部识别。用dlib及face\_recognition两个库来实现获得唇部的特征点。将这些特征点用直线连接，大致获得唇部。然后对这一唇部用opencv进行分析。获取RGB。
3. YUV空间的应用。我们的搜索功能对颜色十分的敏感。而不同的亮度下，同一颜色的RGB会有明显差异。因此我们采用了YUV空间来减少亮度对照片搜索结果的影响。YUV空间的好处在于亮度Y与色度U、V相互独立。同时YUV空间与RGB空间一一对应，满足：

Y = 0.299R + 0.587G + 0.114B

U = -0.147R - 0.289G + 0.436B

V = 0.615R - 0.515G - 0.100B

我们对口红库中的RGB获取其YUV。同时对唇部的RGB得到其YUV，首先忽略亮度，对UV进行比较计算高斯距离。但是，由于缺少Y，导致无法做到一一对应，因此很有可能会出现多个结果，甚至可能是色度相近，亮度相差很大的结果。因此，我们重新对RGB空间计算高斯距离。然后将两个距离加权，得到最后结果。对这一结果进行排序。获得最为相近的商品。

4、口红图片识别，则是运用了SIFT算子来算出其特征向量并存储，之后用暴力搜索的方法来计算匹配点并返回最优结果。

试妆：

试妆的大致思路同图片搜索类似。在获得你想要试的商品并确定唇部之后，先获得商品及唇部的YUV，对新唇部的UV赋予商品的UV，对新唇部的Y赋予嘴唇的Y与商品的Y的加权值。（在尝试后，我们发现将新唇部的Y直接赋予原唇部的Y达到的效果最好。）

最终获得结果

页面设计：

页面在设计的过程中通过code.py将前端与后端，以及所有的功能整合了起来。通过url的不同特点，我们可以实现界面之间的跳转，界面及其操作过程可见视频。

在页面的编写过程中，编码以及页面的跳转使我们所遇到的最大的问题，通常的措施是从源头开始。由于爬取到的文字当中DIOR等产品名字当中可能会有法语，故需单独处理，而页面的跳转通常失败的原因是正则表达式之间的冲突以及底层代码的错误。

页面的美化过程十分繁琐，配图以及文字大小的调整都需要经过反复思考，我们还制作了自己的logo。最终的页面结果可以通过运行code.py来查看。我们先在报告中列出一些截图作为参考：













