

Report of Final Project

Wang ZhongYe

2019 年 1 月 13 日

1 成果展示

我们的搜索引擎采用Bootstrap作为前端的框架，利用Javascript实现了与模型有关的部分的前后端的异步交互，避免页面的加载时间过长。我们的搜索引擎后端利用web.py实现对网页前端的支持，使用高效的elastic-search实现数据的索引和搜索。

我们使用额外数据和词频分析建立了现代诗和古诗的词典和TF-IDF词典，并借助jieba的有关功能实现了现代诗和古诗文的文本分析和关键词抽取。我们使用深度卷积神经网络实现了图片到物象的转化以及建立图片与诗歌间的联系从而实现为诗歌配图、由图片搜索和生成现代诗和古诗文。

首页推荐 图1所示为我们搜索引擎的首页，其中包含了每日自动更新的推荐内容，和每次刷新都会更新的诗歌推荐内容。推荐的具体算法可以通过浏览器的Cookie取回用户的常用搜索关键词进行相似度推荐，或利用用户浏览的诗歌进行关联推荐。由于时间原因，我们未能着手实现推荐算法，目前使用随即取回诗歌填充推荐的内容。



《我等候你》

我等你
我望着户外的昏黄
如同望着将来
我的心疲惫了我的听
你怎还不来希望
在一秒钟上分许开花

— 徐志摩

《城中》

商店之行列永远是年青的
时时闪耀着孩子的眼睛
向每一个过路人作态
若有意若无意

过路人永远是年青的

— 施蛰存

《手推车》

在黄河流过的地域
在无数的枯干了的河底
手推车
以唯一的轮子
发出使阴暗的天穹痉挛的尖音
穿过寒冷与。

— 艾青

诗情画意

《木兰花》

十二阑干衰画箔。
取次穿花成小酌。
彩霓舞罢凤孤飞，回首东风空院落。
杳杳桃源仙路渺。
晴日晓窗红薄...

— 舒婷



《于飞乐》

毛滂
记者脚，浓墨里，一片行云。
不多时，梦破云惊。
听噩梦，声断绝，并庭银瓶。
不知多苦，等闲便，结得同...

你可能对这些标签感兴趣

浮动 母亲 守候 转体 目录 故事 题材
山脚 先教 衣色 作态 窗户 衡元 白色
小村 红光 王华君 于飞 带插 纳房 泥墙
船子 年青 现实 网络 四处 云烧 一番
未多时 对话人 手推车 彩蝶舞 想想 牵挂
过节

《法驾导引》
朝元路，朝元路，同驾玉华君。
千乘载花红一色，人间遥指是祥云。
回望海光新。
— 陈与义

Figure 1: 网站首页

搜索接口 作为一个搜索引擎，其最核心的页面要素便是搜索接口。图2所示为我们搜索引擎提供给用户的接口，该接口在网页所有页面都存在。该搜索表单嵌入在页面的导航栏中，默认情况下收起高级搜索表单，仅显示搜索诗歌的类型（全部、现代诗、古诗文）和模糊搜索输入框，展开后用户可以自定义当前的查询细节。



Figure 2: 搜索接口

首先，我们允许用户选择在哪些域中进行模糊搜索。打开相应的开关可以另当前查询包括相应的域，有多个域被选择后，搜索引擎将返回匹配尽可能多的域的结果。翻译和赏析仅对古诗文搜索类型有效。如果所有域都被关闭，当前查询会被归类为无效查询并反馈相应信息给用户。网页默认开启所有的搜索域。

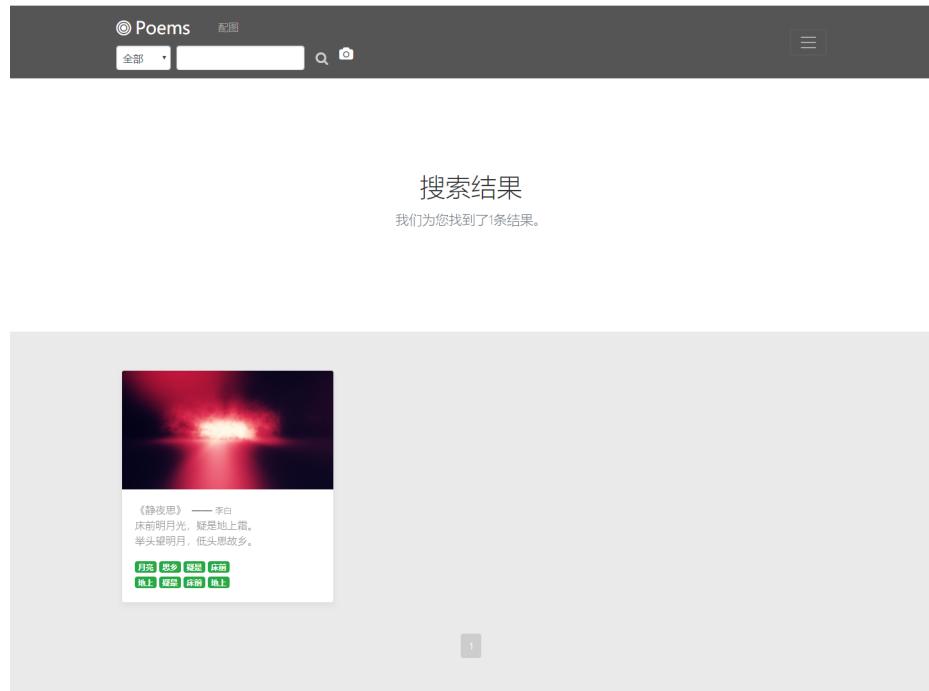
其次，我们提供同义词扩展功能。当用户对查询的关键词不确定时，可以开启这一功能，搜索引擎将会利用内置的词库对查询中的每个词进行同义词扩展，扩大搜索结果的覆盖范围，提高用户找到所希望得到的结果。这一功能仅对在内容中搜索的查询进行扩展。

最后，我们提供精确查询接口，让用户进一步明确当前查询的细节。对于不同的查询类型，我们提供不同的搜索域。对于古诗文，用户可以明确诗歌的标题、作者、标签、朝代或类型（诗、词、曲等）；对于现代诗，用户可以明确诗歌的标题、作者、标签、流派或年代。如果一个域在精确搜索中被使用，该域中的搜索将不会使用模糊搜索框中的字段。

图2中所示的样例查询仅在作者域中模糊搜索“李白”这一字段，不使用同义词扩展，在标题域中精确搜索“静夜思”这一字段。其余字段为空将不对它们进行精确搜索。

除此之外，用户还可点击搜索按钮边上的相机按钮，上传希望分析并用于搜索诗歌的图片。用户还可通过上方的链接跳转至配图页面，上传自创的诗歌并为之匹配合适的图片。

结果页面 图3所示为上述查询的搜索结果，可见搜索引擎精确的返回了一条作者是李白，诗名为《静夜思》的诗歌。



Copyright © 2019-2023.
Poem Inspire is created by Bruce Wang, Mark Dana, Jimmy Li, Apple Chen.
Have fun reading poems!

[Back to top](#)

Figure 3: 精确搜索结果

点击单首诗歌可以跳转到如图4所示的相应的诗歌内容页面。该页面会显示诗歌的完整内容、标签以及相应配图。



Copyright © 2019-2023.
Poem Inspire is created by Bruce Wang, Mark Dana, Jimmy Li, Apple Chen.
Have fun reading poems!

[Back to top](#)

Figure 4: 单首诗歌内容页面

如果解除精确搜索，并在所有域中模糊搜索“李白”这一字段，将返回如图5所示的结果页面。其中显示该查询共有136条匹配结果，并且每页显示一定数量的结果诗歌。

The screenshot displays a search results page for the query "李白". At the top, there's a navigation bar with a logo, a search input field, and a search button. Below the header, a message says "搜索结果" (Search Results) and "我们为您找到了136条结果。" (We found 136 results for you.)

The results are presented in a grid format. Each result card contains:

- A thumbnail image related to the poem.
- The poem title (e.g., 《寻李白》, 《把酒对月歌》, 《赠汪伦》, 《李白传奇》).
- The author's name (e.g., 余光中, 唐寅, 杜甫).
- A brief description of the poem.
- Tags associated with the poem.

At the bottom of the page, there are navigation links for page numbers (1, 2, 3, 4, 后一页, 尾页) and a "Back to top" link.

Figure 5: 模糊搜索结果

对于每首诗歌，我们通过??为其匹配了一张图片，并通过诗歌文本分析的算法结合先前爬取的数据对其添加合适的标签，同诗歌一起展示给用户。如果诗歌长度过长，将会截断一定长度后显示。在电脑浏览器上看，每条结果的大小参差不齐，但我们将是针对手机端开发的网站，在手机端上浏览效果比较好。由于结果数目较多，我们实现了搜索结果的分页显示。

用户也可以浏览单个作者的信息及其所有作品，该页面同搜索结果页面采用相同的实现形式和页面效果，再次不再作图片展示。

诗图转换 除了文本搜索，我们还实现了图片与诗歌间的转化。图6所示为图像分析页面，这里显示了对用户上传的图片的分析结果，用户可以在分析结果的基础上进一步搜索诗歌或者生成诗歌。图中是现代诗生成结果的样例。



Figure 6: 图像分析页面

图7是诗歌配图页面的样例。这里，用户可以上传自己创作的诗歌，并为这首诗歌匹配图片。图中以“青草在奔跑”为例进行配图，结果中以奔跑为主题的图片居多。

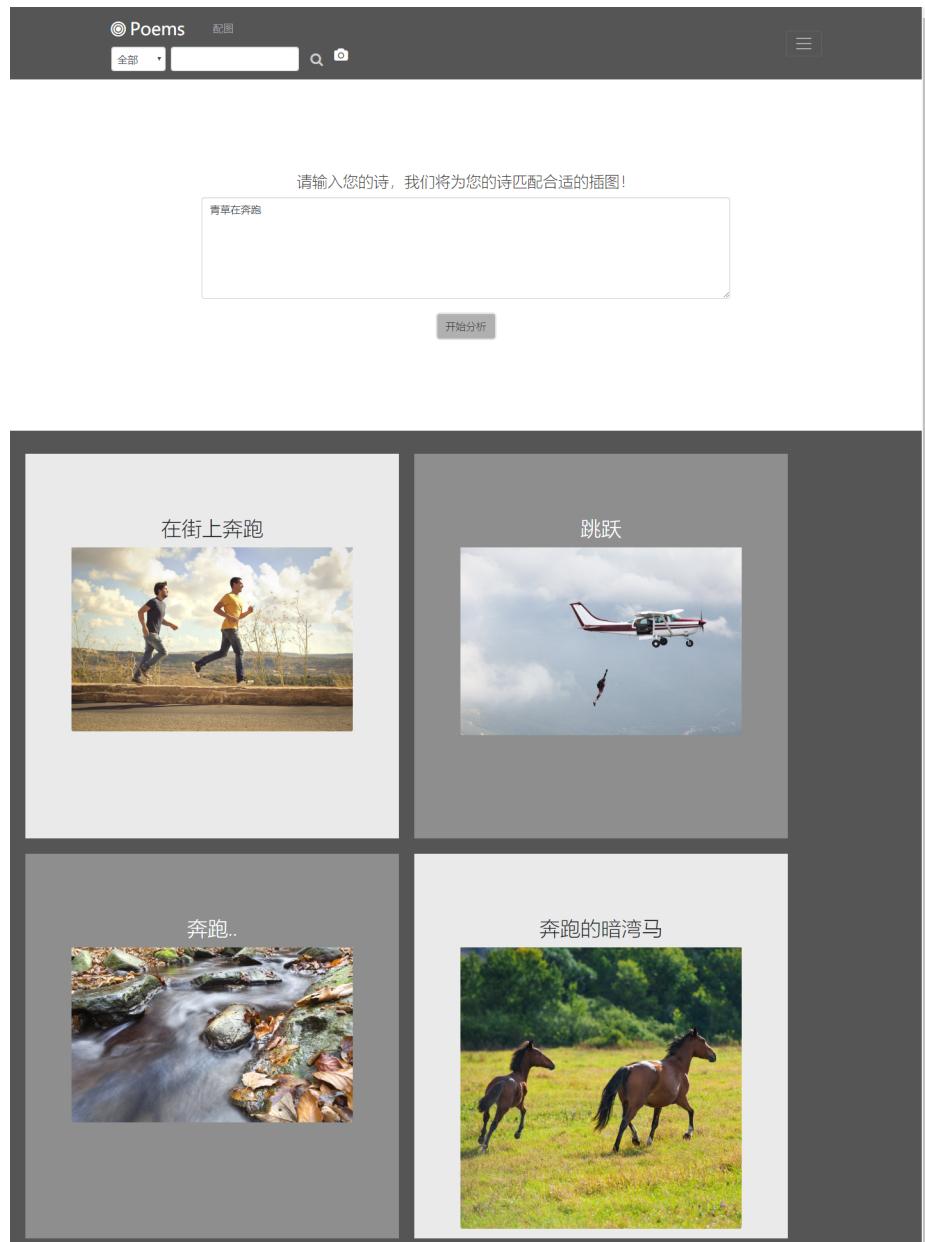


Figure 7: 诗歌配图页面

以上便是我们搜索引擎的简要的展示，接下来我们会详细介绍各部分的实现细节。

2 后端实现

在这一部分，我们将介绍搜索引擎的后端实现，包括网站的后端实现和数据库搜索的实

现。

2.1 网站架构

图8是我们搜索引擎的网站架构。

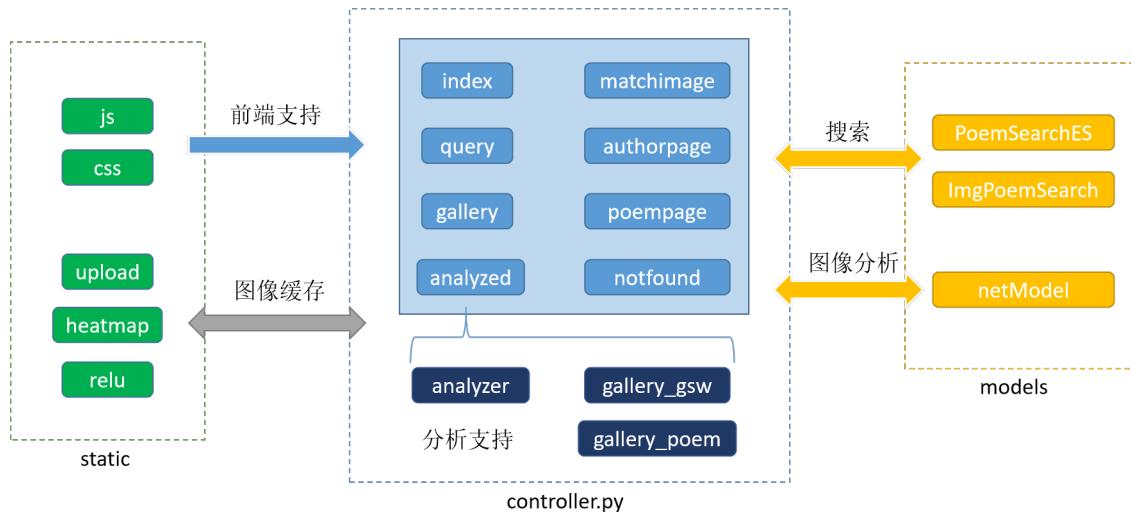


Figure 8: 网站架构

我们的网站采用了一定的MVC分离。PoemSearchES和netModel及model文件夹下的模型为搜索引擎的模型部分，负责对controller发出的查询请求做出响应和进行图像处理。controller.py为我们的控制器，图中由淡蓝色矩形框出的类都有对应的前端页面，深蓝色标出的三个类通过前端的javascript进行动态加载，异步返回数据。static是web.py框架下网站的资源文件夹，其中的js文件夹和css文件夹包含了支持前端布局的js文件和css文件，其余三个文件夹用于缓存用户上传和图像处理中间结果的图片。

类	URL	页面
index	/index	首页推荐
query	/query	文本和图像查询处理结果页面，无翻页功能
gallery	/gallery	同query，但只处理文本查询，提供翻页接口
poempage	/poempage	单首诗歌内容页面
authorpage	/authorpage	单个作者信息及作品页面，类似gallery
matchimage	/matchimage	诗歌配图页面
analyzed	/analyzed	图像分析页面
notfound	/notfound	404页面

Table 1: 链接对应关系

controller中各个类对应的url及前端页面功能如表1所示。

在controller.py中，我们还有一系列辅助函数存放在validator类下，用来进行表单验证和表格输入的预处理。其中最主要的两个函数是form_validate和to_command_dict。

form_validate对于用户通过表单的输入进行验证并返回验证结果供调用者进一步判断页面的跳转。该函数会判断用户的输入是否为空，其会检查所有可能成为有效输入的域，包括高级搜索中的精确搜索域。该函数还会检查表单输入的合法性，来避免用户的恶意访问。两个关键的准则是表单中包含searchType和query这两个域，因为他们确定了查询的索引和查询的内容（虽然query可以为空），和存在query模糊查询时高级搜索中的搜索域选择非空，否则这条查询无法转化成模型可以处理的命令。对于其他只应该由我们规定的内链所引起的查询，我们或者重用了form_validate，或者实现了各自的验证函数来保证没有会造成严重后果的恶意访问发生。

to_command_dict根据表单提供的不同的查询约束，生成对应的查询域和查询值的映射并规定每一条约束是否为强制的（精确查询的）。这个函数的意义在于实现搜索引擎建立与数据库建立、控制器与模型的解耦，避免了在多个文件中修改关键字名称的工作。

2.2 索引搜索

2.2.1 文本搜索

首先，我们将介绍我们是如何利用elastic-search实现基于文本的诗歌搜索的。

布尔查询 我们充分利用了elastic-search的布尔查询的灵活性，从而得以实现复杂的符合查询。从控制器中传入到后端搜索模型的数据都是经过处理的命令字典，其中每个条目的关键字已转换成数据库中的搜索域的关键字，每个条目的值是一个包含查询内容和是否使用了精确查询的布尔值，该字典形如{[关键字] : ([查询内容], [是否精确查询])}。

对应古诗文的关键字有：“author”（作者），“dynasty”（朝代），“label_tokenized”（标签），“title_tokenized”（标题），“text_tokenized”（诗歌正文），“shangxi_tokenized”（诗歌赏析），“yiwen_tokenized”（诗歌翻译）。

对应现代诗的关键字有：“author”（作者），“title_tokenized”（标题），“label_tokenized”（标签），“text_tokenized”（诗歌正文），“genre_key”（流派），“time_key”（年代）。

对于每条查询，我们都准备了一个包含should（选择性出现）和must（必须出现）两个列表的字典作为搜索主体，对于命令字典中的合法条目添加到对应的列表中。如果使用了精确查询则将子查询体加入到must列表，否则加入到should列表。在常规情况下，我们使用match_phrase作为查询方式，这种情况下，进过elastic分词分析后的每个词组必须按顺序完整地匹配上对应域中的一段子字符串才算是一次命中。如果搜索是由同义词扩展、或者图片搜诗发起的，则使用条件较宽松的match方式进行查询。

在查询的时候，我们首先利用elastic的count查询取回总共的匹配数量，再通过函数调用者规定的页数、每页显示数量等确定取回的记录在所有排序后的匹配中的区间，利用search方法取回搜索结果。

搜索结果后处理 搜索结果是以elastic的json放回格式构造的，不便于控制器和前端显示使用，因而在返回结果前调用process_query_results函数对搜索结果列表进行格式化处理。该函数会对诗歌显示长度进行调整，对每条结果的字典关键字进行重命名，配置内链的URL，对其他的显示内容进行适当的调整等。如此一来，我们实现了MVC三者较高程度的解耦，方便后续功能的扩展和调整。其他类型的搜索也采用了类似的方法进行后处理，后续将不再对此赘述。

上述类型的布尔查询主要用在对于诗歌的查询上，这也是我们搜索引擎最主要的查询功能。我们有cnmodern_search（现代诗查询）和gushiwen_search（古诗文查询）两个主要的查询函数，采用上述实现方式。对于mixed_search（混合查询），我们采用每次取回各占显示数量一半的现代诗和古诗文策略，复用以实现的单类型查询函数。这样做唯一的不足在于如果缺少某一种类型的搜索结果，前端只能显示一半数量的结果。但这样做却能保证我们的分页查询正常工作。

分页查询 我们的分页采用from-size的深度分页，及每次查询取回一定排序区间内的结果，这需要反复取回同一段结果并进行排序，效率较低。而且，elastic对于from-size查询方式的查询上限有一定限制，每次最多只能取回10000条结果，我们将这一上限扩大到50000来应对较大的查询量。这并不能从根本上解决问题，我们可以利用elastic的scroll查询方式来实现高效的分页查询，但由于时间问题，我们只能止步于目前的深度分页模式。

作者查询 作者查询并不是我们搜索引擎的核心查询，但也是其中必不可少的一部分，因为我们需要对一个作者的作品进行展示并允许用户通过一首诗了解其作者的其他作品。搜索作者时，我们采用了使用match_phrase命令来匹配作者姓名来搜索结果。但稍后我们发现可以使用elastic数据库的内置ID来准确的取回某个作者的结果，现在的方法对于重名作者将无能为力。由于时间问题，我们未能及时改善此处的查询。get_author_poems函数实现了作者诗歌的分页查询，get_author_desc用于取回单个作者的简介。

除了上述查询，我们模型部分还有基于elastic数据库内置ID的单首诗歌的取回和推荐取回，这部分的查询实现较为简略，在此不多加赘述。

2.2.2 图像查询

3 爬虫

3.1 古诗文网爬取

古诗文网（<https://www.gushiwen.org/>）中包含了中国古代大部分的诗、词、曲、文，并提供了相应的翻译、注释、赏析、标签标注等额外信息。我们希望可以将这部分数据加入到我们的数据库中，实现较完备的搜索功能。

但是，我们发现古诗文网仅有诗歌的内容是通过静态的方式加载的，其余额外信息是通过javascript动态加载的，无法通过BeautifulSoup解析。于是，我们使用selenium模拟浏览器访问，通过模拟鼠标点击相关按钮来取回上述的额外信息。我们使用BeautifulSoup和selenium同时配合爬取，主要利用静态网页下对每首诗歌的ID的信息，来确定对应按钮的ID来实现点击。

虽然使用selenium会大幅降低爬取速度，但由于该网站限制了最大页数为1000，所以总体时间还是可以接受的。

3.1.1 VEER图片数据爬取

VEER (<https://www.veer.com/>) 是一家免版税、国际化的微图提供商，其中提供了大量高质量的风景及其他主题的图片，我们主要使用其中的图片作为搜索引擎的图片数据。其高清图片需要付费购买才能使用，所以我们只使用了低质量的缩略图，并给出了原始VEER图片链接。

再爬取该网站的时候，我们发现Urllib和BeautifulSoup无法解析其源码，原因是该网站存在反爬虫机制，会对HTML源码注入使爬虫失效的内容。针对这种机制，我们采取了利用selenium调用javascript来强制返回其完整源码的方法。为了最大限度地降低selenium爬取时间，我们采用了先下载全部页面再解析的方式，所幸该网站可以通过URL来设定单页图片显示数量（很不明智的设计，但其后端限制了最大返回数量为200），我们通过设置显示数量为200的方式降低了selenium的调用次数，从而减少爬取时间。

我们对其中的“风光”关键字的图片进行了爬取，总计约18万张图片。