

大作业报告——诗歌搜索引擎

戴昊悦 王中烨 李竟宇 陈浩平

2019 年 1 月 13 日

1 项目综述

读诗是一种美好的享受。但目前网络上的诗歌库，比如中国诗歌网、古诗文网等，要么是数据量较小，要么是表单样式的网站，没有丰富的功能。我们希望将诗歌用一种更好的形式展现给用户，因此利用本学期电工导知识实现了此诗歌引擎：Poem Inspire。

该引擎数据量涵盖了全部诗经、唐宋古诗 30 万余首、宋词元曲等 2 万余首、中文现代诗 5 千余首、英文现代诗 1 万余首、诗歌配图约 20 万张、赏析 5 千余篇，以及相应的 2 万余名诗人信息、1 万余个标签。

我们实现的功能包括：

- **信息整合：**对每首诗提供信息页面，包括正文、作者、意象、标签、年代、自动配图，有助于读者对一首诗产生直观印象；
- **分类查看：**对作者、流派、标签等制作专门页面，方便用户阅读研究该类别下的诗歌；
- **模糊搜索：**将用户的输入对标题、正文、赏析等多个搜索域进行搜索，展现全面的内容；
- **精确搜索：**集成在在高级搜索的多个选择框内，用户可指定对某类诗的某个域搜索；
- **词义联想：**考虑到用户输入现代汉语与古诗词的差异性，我们实现了古词联想，比如输入“酒店”，联想到“逆旅酒楼客舍酒肆旅亭馆驿帆宿客馆酒家炊烟厨香杏花村...”；
- **自动配图：**用户输入某首诗，将分析其表达的意象，并返回主题最适合的配图；
- **以图搜诗：**用户上传一张图片，将分析图片包含的物体、场景、情感信息，搜索返回与图片最相符合的诗歌；
- **以图写诗：**用户上传一张图片，并可以自由选择特征信息，自动从图片生成现代诗、绝句及律诗。
- **每日推荐：**主页每日推荐，包括横幅诗图、古诗现代诗推荐、标签推荐等；

2 信息获取及整理

项目诗歌来源包括中国诗歌网、诗歌大全文库、古诗文网等，包括现代诗与古诗这两大门类。图片来源除了上述中国诗歌网的诗影栏目，还有 veer 图库。

2.1 信息获取——爬虫

静态网页 以中国诗歌网为例，该网站没有反爬虫机制，可以进行比较方便的爬取。网站目录中一页有 60 个现代诗的网址。注意到网站信息包含诗的 ID 信息体现在网站的 URL 中，但 ID 是随机跳跃的递增序列，所以我们无法直接根据 URL 的信息来获取所有网站。我们一开始采取的方法是先选取一个现代诗的页面作为种子，然后进入该页面根据其“下一页”的链接作为爬取的下一个网站。但这样的做法有一个很大的缺陷是爬虫的鲁棒性太低，因为一个页面最多只能生成下一个页面的链接，遇到错误时程序就会终止。最后采取的方案是，根据目录中的页数，每 10 页选择 1 个种子，并使用多线程进行爬取，每次爬取依然选择使用用“下一页”获取下一个链接放入队列，并设置每次爬取的最大页面数为 5000。由于队列先进先出，我们对队列依次出队的操作可以大致保证每个初始种子所引出的页面数大致相当，这样我们一次爬取的 5000 个网站中便能保证其中的页面不会有重复。由于该网站数据量较大，我们对上述程序做了分布式处理，可以在多台电脑上同时爬取，提高速度。

动态网页

2.2 信息整理

网页信息提取 以诗影为例，它是中国诗歌网的一个栏目，它包含了整理好的现代诗及其配图。每个页面都包含一组现代诗及其对应的一张图片，页面结尾处注明了该组诗歌的作者信息。通过分析网站结构可知，每首诗第一行为标题，其后为正文，然后接一张图片。在页面开头有作者姓名和收藏数，收藏数可作为诗影文章受欢迎程度的依据。综上，先爬取原网页，再通过 BeautifulSoup 提取每首现代诗的标题、作者、正文、配图地址和收藏数这些信息，并将这些信息导出为 json 文件，方便后续处理。古诗文网中的古诗除了上述信息，还有朝代、标签、译文、赏析等信息，均提取出来以备存。

多源信息整合 现代诗的来源有多个，包括中国诗歌网的两个栏目和诗歌大全文库，将两部分来源的数据整合为统一的 json，综合其标题、作者、正文、配图等信息，古诗文还需考虑朝代、诗体等。作者信息也需抽取出来单独做一个数据文件，方便后续索引。

3 建立索引

我们初始用 lucene 建立索引，之后由于我们更大量数据与更深层次的搜索需求，我们放弃 lucene，而选用 ElasticSearch。

3.1 Lucene 的尝试

环境：lucene 4.9.0

Lucene 的索引建立是基于结巴分词，将内容、标题（以及关于古诗文的赏析、译文）进行分词，并将其和诗歌的其他信息一起，根据需要选择是否存储与建立倒排索引。由此对应的索引搜

索将诗歌的内容作为必须匹配项，而将标题、作者、朝代信息作为“或”逻辑匹配项进行组合搜索，由此可以提升作者想要搜索内容在搜索结果页面的次序。

3.2 ES 建立最终索引

环境：ElasticSearch 6.5.4

3.2.1 初步索引

使用 elasticsearch 建立索引，首先需要定义索引的映射，及文档中不同字段的名称和属性。这里使用了两种类型：keyword（关键词类型）和 text（文本类型）。keyword 指不进行分词，仅能根据精确值进行查找的字段；text 指进行分词，可以由不同搜索算法进行搜索的字段。以现代诗为例，它包含了标题、作者、正文、配图等信息。配图地址采用 keyword，其他采用 text。作者字段可能包含多个作者，之前整合数据时进行过处理，使作者间用空格相隔，所以用 whitespace 分词器进行索引和搜索。标题和正文各使用两个字段，一个保存原文（如 text），另一个保存进行过分词处理的原文（如 text_tokenized）。索引分词采用 ik_max_word 分词器，它对文本做最为细粒度的拆分，即穷尽所有可能的组合进行分词。搜索分词采用 ik_smart 分词器，进行最为粗粒度地拆分。具体建索引时，使用 bulk 批量操作，提高导入索引数据的效率。

古诗文索引与现代诗索引类似，只是字段数量更多。作者信息包括姓名、简介等，索引方式与上述建立方法类似。

3.2.2 更新索引

在索引建立完成之后根据我们之前重新获取的作者信息以及生成的新标签，我们需要对原有索引进行更新。更新主要包括：现代诗作者的信息扩充（流派、年代），古现代作者的索引整合，添加诗歌标签等。

对于现代诗作者流派、年代信息的添加，我们先将所有作者添加三个空项：time_text, time_key, genre 分别表示年代原文（e.g. “20 世纪 30 年代”），年代关键词（e.g. “20-30”），流派（e.g. “七月派”），然后再根据我们获取的数据将 json 转换为字典，运用 update_by_query() 方式，对作者姓名做匹配，并添加相应信息。

对于古现代作者的整合，我们将古代、现代作者合并到一个 author 索引中，并新增一个字段来区分古代、现代作者。

对于添加诗歌标签，我们主要运用 jieba.analyse.extract_tags() 方法。对于现代诗歌，因为其已经与我们日常用语很接近，所以我们直接用结巴的默认库，对其提取最相关联的 5 个标签。我们限定我们提取的标签为名词（“n”）以及地名（“ns”），因为这些词语一般能包含更加具体和重要的信息。对于古代诗歌，我们仍然是基于 tf-idf 算法进行关键词抽取，用 set_dictionary() 的方法将主词典设置为古词词典（“guci_dict.txt”），并设置频率文件：jieba.analyse.TFIDF(idf_path='guci_idf.txt')，频率文件为在我们数据库里的诗歌中所有词语所出现的频率，我们运用 jieba.analyse.extract_tags() 方法选取前三个匹配，并且添加到原来古诗标签字段后（原来的标签是基于诗歌的分类，比如

“唐诗”、“给孩子的诗”等)。最后，我们用 `update_by_query()` 的方法，按诗歌 ID 对古诗和现代诗分别添加和修改标签项。

3.2.3 给诗配图

我们首先确定了基于关键词-标签搜索的诗图互搜算法，算法原理如图1所示。

- (1) 由诗搜图的过程是由诗的关键词进行同义词扩展后搜索图片的联想词。若由诗的全文分词进行搜索，则查询信息数据量较大，且缺乏有效信息。而关键词通过结巴的 tf-idf 算法进行抽取，其本身是文章重要信息的提炼，查询有效性提高。
- (2) 由图搜诗是由图的联想词搜索诗歌的全文。图片的联想词已经进行过同义词古词扩展，搜索全文(分词后的)可以匹配更多的信息，对古诗文部分也有所兼容。

根据上述诗图互搜的两种方式，诗配图也有两种方案。

- 方案一：遍历诗歌，通过诗歌查询图片，将评价最高的图片作为其匹配图。
- 方案二：遍历图片，通过图片查询诗歌，在评价得分最靠前的五首诗中，选择未配图的第一首诗作为该图的对应。

数据库中图片有 15 万张，而诗歌数只有 3 万。经过测试，第一种方案速度约为 1 秒 5 首诗，第二种约为 3 秒一张图，所以第一种方案速度更快。两种方案均存在诗歌未匹配上图片的情况，后一方案以图片为主体的搜索使这一情况更容易发生，因此还需采用前者对剩余诗歌进行配图，增加了工作量。综合考虑，我们采用了第一种方案。

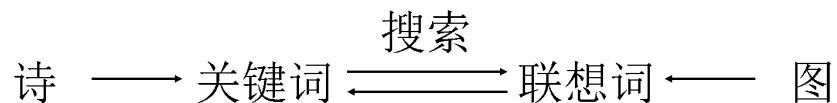


图 1: 诗图互搜算法原理图

4 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。

4.1 网站架构

图2是我们搜索引擎的网站架构。

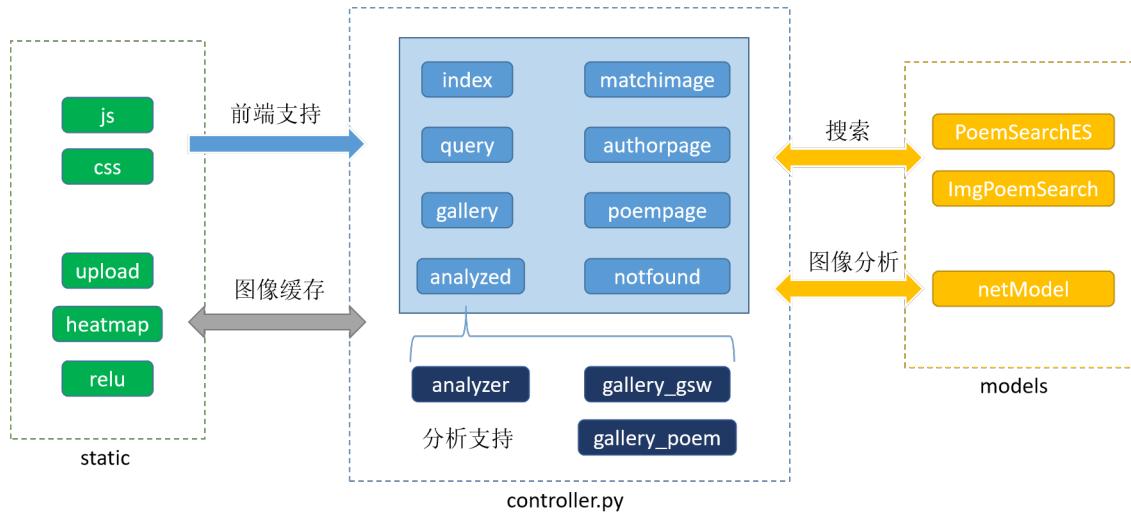


图 2: 网站架构

我们的网站采用了一定的 MVC 分离。PoemSeachES 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

表 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。

4.2 索引搜索

4.2.1 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结果页数、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，因而我们在返回结果前调用 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。

4.2.2 图片搜索

图片搜索与前文给诗配图的原理一致，如图所示。

5 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3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，目前使用随机取回诗歌填充推荐的内容。对于每日推荐，将当天日期作为随机数种子，定义相应变量保存诗歌 id 信息，每天更新一次。每日一诗由于位于页首，背景为其配图，限于篇幅，对其正文长度限制在 50 字以内，对不符合要求的诗歌进行线性搜索。页面最后列出了这些诗歌对应的标签，方便用户点击查询。



《我等候你》

我等你
我望着窗户的昏黄
如同望着将来
我的心震颤了我的听
你怎还不来希望
在一秒钟上允许开花

—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
若有意者无意
过路人永远是年青的

—南星

《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
发出使阴暗的天空多寥的尖音
穿过寒冷与...

—艾青

诗情画意

《木兰花》

十二阑干春匝泊。
取次穿花成小酌。
彩鸾舞罢风孤飞，回首东风空院落。
杳杳桃源仙路渺。
晴日晓窗红薄...

—舒��



《于飞乐》

王维
记曾携，浓睡里，一片行云。
太多时，梦破云惊。
听辘轳，声断也，并底恨焦。
不知罗带，等闲便、结南园...

你可能对这些标签感兴趣

浮动	母亲	守候	转铃	归宿	依次	同鸣
山脚	先教	表色	作态	绝世	朝元	红色
小村	灯光	玉华君	黑屏	生趣	物伤	浓郁
帖子	年青	蝴蝶	画面	闻此声	云极	一册
未多时	过路人	手推车	彩蝶舞	版图	妙理	
红薄						

《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

图 3: 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4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。



图 4: 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。

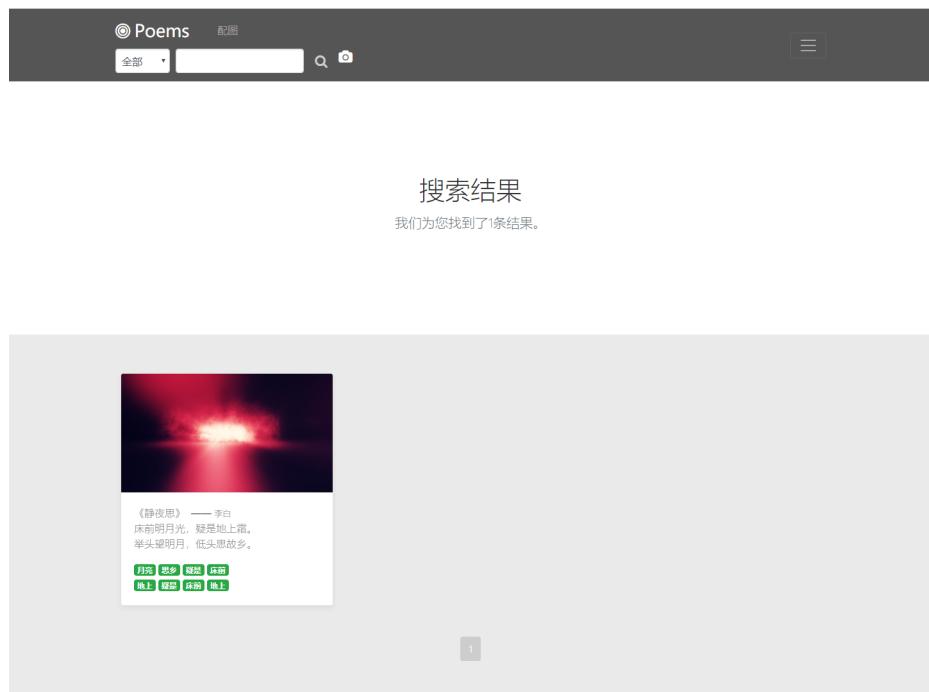
其次，我们提供同义词扩展功能。当用户对查询的关键词不确定时，可以开启这一功能，搜索引擎将会利用内置的词库对查询中的每个词进行同义词扩展，扩大搜索结果的覆盖范围，提高用户找到所希望得到的结果。这一功能仅对在内容中搜索的查询进行扩展。

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4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5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](#)

图 5: 精确搜索结果

点击单首诗歌可以跳转到如图6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](#)

图 6: 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7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。

© Poems 配图 全部

搜索结果

我们为您找到了136条结果。



《寻李白》 —— 余光中
痛饮狂歌空度日
飞扬跋扈为谁雄
那一双傲慢的靴子至今还落在
高力士羞愧的手掌里却不见了
把满地的难民...

写人 酒肉 飞扬 跋扈
李白 七绝 奔腾 气势



《把酒对月歌》 —— 唐寅
李白前时有明月，惟有李白诗能说。
李白如今已仙去，月在青天几圆缺？
今人犹唱李白诗，明月还如李白时。...

写人 对月 把酒 李白
李白 月夜 仙人



《戏李白》 —— 余光中
你曾是黄河之水天上来
阴山动
龙门开
而今反从你的句中来
惊涛与豪笑
万里涛声入海
那轰动寰宇的大潮...

黄河 李白 余光中
李白 黄河 苏东坡 多才 波涛



《赠汪伦》 —— 李白
李白乘舟将欲行，忽闻岸上踏歌声。
桃花潭水深千尺，不及汪伦送我情。

写人 酒肉 友情 送别
李白 汪伦 行路



《李白传奇》 —— 崔夫
相传峨眉峰顶有一块巨石上刻有一张白纸一天午
风大雨作天震地撼之际一只硕大无比的鹏
鸟碎石破纸冲天而...

传说 神话 峨眉 风雨
李白 飞腾



《赠李白》 —— 杜甫
秋来相顾尚飘蓬，未就丹砂愧葛洪。
痛饮狂歌空度日，飞扬跋扈为谁雄。

杜甫 李白 丹砂 葛洪
李白 飞扬 跋扈

1 2 3 4 后一页 尾页

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

Back to top

图 7: 模糊搜索结果

12

对于每首诗歌，我们通过关键词-标签搜索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8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。



图 8: 图像分析页面

图9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。



图 9: 诗歌配图页面

以上便是我们搜索引擎的简要的展示。