# 古诗文推荐系统

王胜广、龚润宇、梅楚鹤、柳俊志、尹国健

**摘要：**诗词推荐属于文本推荐，但诗词的特点与电影有很大相似性，其生命周期较长。诗词作为一种情感表达的载体，反映的是诗人当时的情感。对诗词共鸣则取决于读者此时此刻的心情和所处环境。在很大程度上受到位置、时间、天气等上下文信息的影响。故诗词推荐主要基于上下文相关信息进行。文中首先给出推荐系统的总体框架；然后对读者和诗歌的位置、时间、季节、天气、节日等上下文信息进行挖掘，然后进行推荐。基于上下文的古诗文推荐系统也可以是本文的标题。

**关键词：**诗词推荐；推荐系统；上下文；

“上下文”通常的定义是指所有可以表征物体特征的信息。在位置、物体属性、时间等维度都存在丰富的用户及被推荐对象的上下文信息[[4](javascript:void(0);)]。比如，用户的上下文信息通常是由不同的属性组成：物理位置、情绪状态、个人历史、天气因素等。在个性化推荐系统中，将上下文信息融入进来将有效地提高推荐精确度和用户满意度。

在我们的诗词推荐系统中，通过开源诗词数据库、web爬虫爬取的诗词中挖掘的信息构建诗词信息库，利用天气信息、ip等用户上下文信息建模。根据挖掘的用户上下文和诗词标签，产生个性化的推荐结果。

推荐系统构建分为以下4个阶段：（1）数据收集与预处理：通过相关技术获取用户作用于项目的相关历史行为数据并按所需规整数据；（2）数据挖掘：诗词诗人数据和读者的隐含上下文信息；（3）推荐列表生成：根据诗词标签，结合获取的上下文信息为用户提供推荐列表；（4）反馈：根据相关评价方法和评价标准对推荐列表中的结果进行评估，并做出适当的改进。下面将详细介绍数据挖掘及推荐系统的实现：

## 诗词数据挖掘

本次用到的数据来源于古诗文网<https://www.gushiwen.org/> ，共包含从先秦到现代的诗词共66105首，主要为唐宋两个时期的诗词，其中唐代42491首，宋代19736首，元代1391首，先秦659首。诗人3150位，其中唐代1599人，宋代843人，元代240人。诗词数据主要为唐诗宋词，唐诗雄壮浑厚、气势开阔，宋词清新婉约、精微深细。每首诗词大概包涵诗词背景、诗词内容、诗词朝代、诗词翻译、诗词名词、诗词作者、诗词赏析、诗词标签等信息。标签信息是我们主要需要的信息，现有标签1000个，包括诗词分类（如唐诗三百首），季节（春夏秋冬）、情感（思乡、爱情、离别）、节日（冬至、重阳）、写物（松、竹、梅、菊）。

为了结合用户信息进行精确地诗词推荐，我们对诗词的地点、季节、节日、写物、天气、时间等信息进行了挖掘。将现有标签分为4类：季节（春夏秋冬）、情感（思乡、爱情）、节日（冬至、重阳）、写物（松、竹、梅、菊）。增加3类：时间（朝、暮）、地点（琅琊、石头城）、情绪（悲、惧、乐、怒、思、喜）。

为了给大多数诗词加上标签，我们将标签作为关键字，在诗词相关字段：about、fanyi、shangxi、content中匹配字串，一共有5万多首诗都被添加上新的标签。此外通过数据挖掘和分析，我们人工选出一些高度相关的词，也作为诗词打标签的依据，主要有一下几种方法：

**（1）.全局高频字**

如下做简单的词频统计。结果如下，“人”字排行第一。而后续的“山”“风”“月”“日”“天”“云”“春”等都是在写景的诗句里经常出现的意象。

然而，对于古汉语（文言文），尤其是诗词的分词处理可没有这么简单，因为单字词占古汉语词汇统计信息的80％以上，再加上古汉语微言大义，字字千钧，所以针对现代汉语的分词技术往往不适用于它。鉴于此种情况，作者采取的是逐字切分的处理方式，同时去掉一些常见的虚词，如“之”、“乎”、“者”、“也”、“不”、“一”等。并且还去掉了一些特殊符号，如“，”、“。”等。

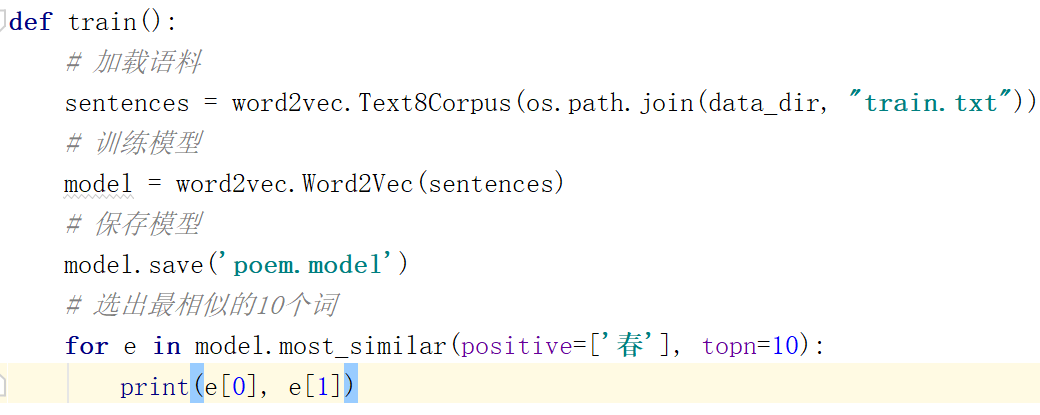
部分高频词展示：

|  |  |
| --- | --- |
| 字 | 字频 |
| 人 | 68515 |
| 风 | 38934 |
| 日 | 36669 |
| 山 | 36069 |
| 云 | 35785 |
| 花 | 34906 |
| 天 | 32688 |
| 月 | 27009 |
| 将 | 18336 |
| 秋 | 18020 |
| 君 | 17893 |

**（2）. 词向量分析**：基于的Word2vec的关联字分析

基于Word2vec的词向量能从大量未标注的普通文本数据中无监督地学习到词向量，而且这些字向量包含了字与字之间的语义关系，正如现实世界中的“物以类聚，类以群分”一样，一个词可以由它们身边的词来定义。

从原理上讲，基于字嵌入的Word2vec是指把一个维数为所有字的数量的高维空间嵌入到一个维数低得多的连续向量空间中，每个单字被映射为实数域上的向量。把每个单字变成一个向量，目的还是为了方便计算，比如“求单字A的同义字”，就可以通过“求与单字A在cos距离下最相似的向量”来做到。

词向量能从大量未标注的普通文本数据中无监督地学习到词向量，而且这些词向量包含了词之间的语义关系，正如现实世界中的“物以类聚，类以群分”一样，字词可以由它们身边的字来定义。我们用jieba分词对文本进行切分，然后使用genism Word2vec训练得到词向量。

下面，选取一些单字进行字向量关联分析，展示如下：

|  |  |
| --- | --- |
| 字 | 关联字 |
| 春 | 清明，时候，春事，芳菲，寒食，花，春色，春光，风光，东风 |
| 思乡 | 望乡，邮亭，乡思，消魂，海边，凄然，别情，迢遥，上楼，落花流水 |
| 梅 | 春思，春晚，早梅，西园，春景，桃，咏柳，赏春，如梦令，闺思 |
| 冬 | 九月，四月，早春，大作，闰，十月，节过，寄王，晦日，西亭 |
| 中秋 | 南楼，重阳，九日，登高，月夜，秋夜，暮春，江楼，元宵，湖上 |

可以看出还是有很高的相关度，我们手动挑选其中的一下词作为关键词给诗词打标签。

我们进一步进行了多维情绪分析，为了丰富分析维度，不采用简单的二元分析，即“积极”和“消极”2种情绪，而是7种细颗粒的情绪分类，即悲、惧、乐、怒、思、喜、忧。

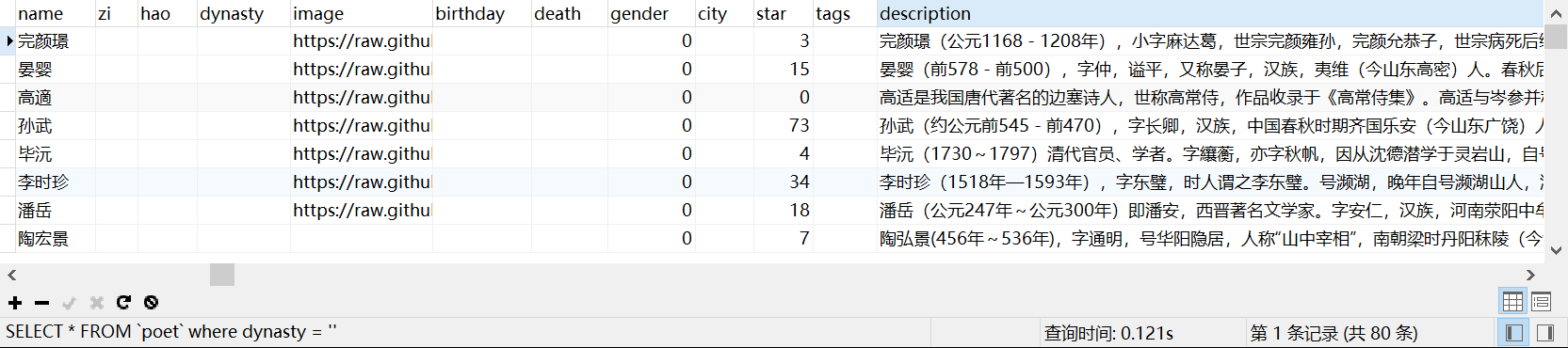
根据上面获取到的字向量，经过人工遴选后，得到可以用于训练的“情绪字典”，根据诗歌中常见的主题类别，情绪类别分为：

|  |  |  |
| --- | --- | --- |
| 标签类型 | 标签 | 关键词 |
| 情绪 | 悲 | 愁、恸、痛、寡、哀、伤、嗟… |
| 惧 | 谗、谤、患、罪、诈、惧、诬… |
| 乐 | 悦、欣、乐、怡、洽、畅、愉… |
| 怒 | 怒、雷、吼、霆、霹、猛、轰… |
| 思 | 思、忆、怀、恨、吟、逢、期… |
| 喜 | 喜、健、倩、贺、好、良、善… |
| 忧 | 恤、忧、痾、虑、艰、遑、厄… |

根据以上的得到的标签关键字，在诗词数据的背景、赏析、内容、翻译中匹配这些关键字，给诗词加上对应标签，我们给大多数诗词都至少打了一个标签。

## 诗人数据挖掘

诗人数据中有用的主要包括名字、朝代、出生地、性别等信息。现有数据集description中会有对诗人的描述，这其中包含了朝代、出生地和性别的相关信息。但是描述的格式并不完全统一，考虑使用正则来尽可能匹配对应的信息，将他们填回到相应的字段。

朝代缺失的有80条。用jieba进行分词，用空格将分词后的结果连接成字符串。之后用正则匹配‘\_朝’或‘\_代’，获得朝代信息，特殊处理‘春秋’时期，之后人工校验了明显错误的部分。

出生地的字段都是没有的。观察描述中对于出生地的表述一般是‘[古代地名]（今[现代地名]）’或者‘[古代/现代地名]人’，因此按照这样的规则进行匹配提取出生地。有一些特殊处理，主要是一些符合规则的其他信息，尽量做了判断跳过。匹配后的结果进行了人工校验，将明显不是地名的结果进行了删除。总共找到3000个诗人的出生地信息。

性别的部分直接在数据库进行了update，使用模糊查找描述中包含‘女’字和‘妓’字的记录，将他们的gender置为1，表示女性，其余记录为0，表示男性。这两类记录有170条，大致进行了人工校验。

UPDATE poet SET gender=1 WHERE description LIKE '%女%';

## 读者上下文信息挖掘

冷启动阶段能获得的用户信息主要的用户ip、访问时间、访问设备等信息。

通过高德地图提供的接口查询到ip所在地址省份和城市，再查询当前的天气情况，结合当前时间。这样我们就得到地点、天气、时间、季节、节日等标签。还可以做历史上的今天等诸多扩展。充分挖掘用户上下文信息。

对上下文信息触发各类标签的条件整理如下：

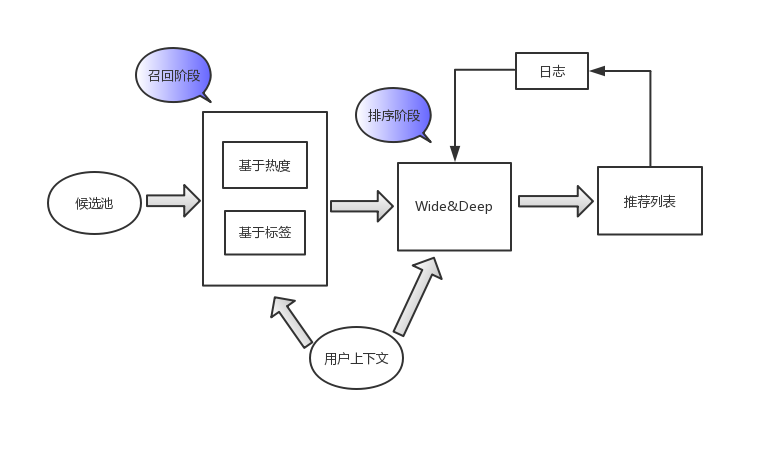
|  |  |  |
| --- | --- | --- |
| 标签类型 | 标签 | 条件 |
| 天气 | 晴 | 天气晴 |
| 雨 | 天气雨 |
| 雾 | 天气雾/ 能见度 |
| 小雨 | 天气小雨/ 降水量 |
| 大雨 | 天气大雨/ 降水量 |
| 阴 | 天气阴 |
| 雪 | 天气雪 |
| 云 | 天气多云 |
| 雷电 | 天气雷电 |
| 干旱 | 天气干旱 |
| 沙尘 | 天气沙尘 |
| 风力 | 风 | 风力 >=2级 |
| 无风 | 风力 <= 1级 |
| 东风 | 东风，风力 >=2级 |
| 南风 | 南风，风力 >=2级 |
| 西风 | 西风，风力 >=2级 |
| 北风 | 北风，风力 >=2级 |
| 微风 | 风力 [2,3] 级 |
| 大风 | 风力 >= 4级 |
| 温度 | 寒冷 | 气温 <= 10 度 |
| 炎热 | 气温 >= 30 度 |
| 时间 | 日出 | 当地日出时间后1小时 |
| 日落 | 当地日落前后1小时 |
| 正午 | 北京时间11-13点（人们通常认为12点为正午） |
| 上午 | [8-11]点 |
| 下午 | [13-17]点 |
| 晚上 | 日落后1小时 - 晚上12点 |
| 凌晨 | [0-5]点 |
| 日期 | 春 | 农历2-4月 |
| 夏 | 农历5-7月 |
| 秋 | 农历8-10月 |
| 冬 | 农历10-1月 |
| 除夕 | 除夕 |
| 春节 | 春节 |
| 新年 | 元旦，春节 |
| 元宵 | 元宵 |
| 寒食 | 寒食 |
| 清明 | 清明 |
| 端午 | 端午 |
| 七夕 | 七夕 |
| 爱情 | 七夕, 元宵, 情人节，白色情人节 |
| 中秋 | 中秋 |
| 重阳 | 重阳 |
| 劳动 | 劳动节 |
| 爱国 | 国庆节，七七，九一八 |
| 妇女 | 妇女节 |
| 母亲 | 母亲节 |
| 父亲 | 父亲节 |
| 儿童 | 儿童节 |
| 老师 | 教师节 |
| 地理 | 华东 | 山东、江苏、安徽、浙江、福建、上海 |
| 华南 | 广东、广西、海南 |
| 华中 | 湖北、湖南、河南、江西 |
| 华北 | 北京、天津、河北、山西、内蒙古 |
| 西北 | 宁夏、新疆、青海、陕西、甘肃 |
| 西南 | 四川、云南、贵州、西藏、重庆 |
| 东北 | 辽宁、吉林、黑龙江 |
| 江南 | 上海、浙江、江苏 |
| 边塞 | 内蒙古，甘肃，宁夏，青海，青海 |
| 西域 | 新疆 |
| 徽州 | 安徽黄山 |
| 长安 | 陕西西安 |
| 武陵 | 湖南常德 |
| 浔阳 | 江西九江 |
| 姑苏 | 江苏苏州 |
| 苏州 | 江苏苏州 |
| 扬州 | 江苏扬州,江苏泰州,江苏南通,江苏盐城,江苏镇江,江苏南京,安徽天长,江苏淮安 |
| 燕京 | 北京 |
| 庐州 | 安徽合肥 |
| 琅琊 | 山东临沂 |
| 石头城 | 江苏南京 |
| 景德镇 | 江西景德镇 |
| 京口 | 江苏镇江 |
| 临安 | 浙江杭州 |
| 广陵 | 江苏扬州 |
| 武陵 | 湖南常德 |
| 钱塘 | 浙江杭州 |
| 金陵 | 江苏南京 |
| 幽州 | 北京 |
| 洛阳 | 洛阳 |
| 凉州 | 甘肃武威 |
| 齐州 | 山东济南 |
| 蜀地 | 四川 |
| 汝南 | 河南驻马店 |
| 大梁 | 河南开封 |
| 泰山 | 山东泰安 |
| 华山 | 陕西渭南 |
| 衡山 | 湖南衡阳 |
| 恒山 | 山西大同 |
| 嵩山 | 河南登封 |
| 黄山 | 安徽黄山 |
| 庐山 | 江西九江 |
| 雁荡山 | 浙江温州 |
| 长江 | 四川攀枝花，四川宜宾，四川泸州，重庆，湖北宜昌，湖北荆州，湖北岳阳，湖北武汉，湖北鄂州，湖北黄石，江西九江，安徽安庆，安徽铜陵，安徽芜湖，安徽马鞍山，江苏南京，江苏镇江，江苏南通，上海 |
| 黄河 | 甘肃兰州，甘肃白银、宁夏石嘴山，内蒙古乌海，内蒙古巴彦淖尔, 内蒙古包头，陕西韩城，山西河津，山西永济、河南开封，河南三门峡，河南洛阳、山东滨州，山东济南 |
| 黄鹤楼 | 湖北武昌 |
| 滕王阁 | 江西南昌 |
| 岳阳楼 | 湖南岳阳 |
| 玉门 | 甘肃敦煌 |
| 阳关 | 甘肃敦煌 |
| 瓜州 | 江苏扬州 |
| 锦城 | 四川成都 |
| 成都 | 四川成都 |
| 洞庭 | 湖南岳阳 |
| 西湖 | 浙江杭州 |
| 赤壁 | 赤壁 |
| 荒漠 | 荒漠 |
| 草原 | 草原 |
| 雪山 | 雪山 |
| 混合 | 梅雨 | 华东地区，6月中-7月中，下雨 |
| 海棠 | 山东，陕西，湖北，江西，安徽，江苏，浙江，广东，广西；4-5月 |
| 桃花 | 华南、西南：2-4月、华中华东：3-4月、华北：4-5月 |
| 牡丹 | 全国；5月 |
| 丁香 | 西南，西北，华北，东北；4-5月 |
| 杏花 | 华中，华东，华北，东北，西南，西北 3 - 4月 |
| 梅花 | 全国 1-3月 |
| 杜鹃 | 华南，华中，华东；4-6月 |
| 荷花 | 全国；6-9月 |
| 桂花 | 全国；6-9月 |
| 桂花 | 华南，华中，华东 9-10 月 |
| 梨花 | 梨花 |
| 菊花 | 菊花 |
| 竹 | 竹 |
| 柳 | 华北 4月 |

## 4．推荐系统具体实现

推荐系统一般分召回和排序两个阶段。召回可以理解为向用户粗选一批待推荐的商品，相当于粗排序。之后会加一层CTR预估的rank模型，相当于精排序。其中召回阶段从数万诗词中召回几百首，排序阶段对召回的诗词精细打分取出top-k返回。我们把地理位置和时间等当成诗词标签，召回阶段：按标签召回和按热度召回，排序阶段使用Wide&Deep模型排序，模型训练方式为离线训练。

总体流程：

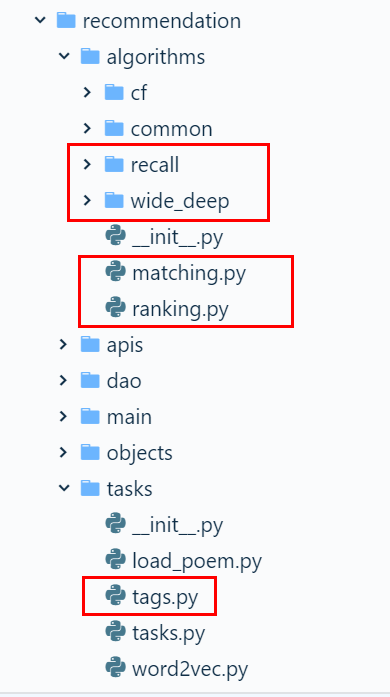
1. 用户上下文：用户访问，获得ip,时间，进行信息挖掘转化为相关标签。
2. 召回：根据（1）的上下文信息，按标签召回+按热度召回，得到一批候选列表
3. 排序：用户上下文和召回候选列表作为输入，得到打分
4. 返回推荐结果并将推荐记录写入日志

下图为推荐系统流程图：

因为缺乏相关数据集进行训练，为了推荐系统能有一定效果，目前推荐系统主要按照召回阶段的结果进行推荐，排序阶段只是为了系统完整性，加入其中，可正常参与整个流程，但最后打分无效。下面将详述推荐系统各阶段所用方法，而不对具体代码实现做讲解，具体代码实现将放在github，并对api文档、部署方式和代码整体架构等配有详细说明文档。

代码地址：<https://github.com/WangShengguang/recommender-system>

下面为部分目录说明，详细说明见github



### 4.1 召回算法

召回部分：按热度召回，按标签上下文召回。把时间和地理位置等全部当成标签。

1.按热度召回：直接按照诗词的点赞数star从高到低排序，返回前k首诗词。

2.按标签召回：用户的ip，访问时间等通过上面的上下文信息挖掘转换为一系列标签，与诗词标签匹配，按照匹配的标签数目从高到低排序，取出前k个。

### 4.2 排序算法

排序部分使用Wide&Deep模型。Wide and deep 模型是 TensorFlow 在 2016 年 6 月左右发布的一类用于分类和回归的模型，并应用到了 Google Play 的应用推荐中。推荐系统可以被看做是一个搜索排序系统，其中输入的query是一系列的用户和文本信息，输出是items的排序列表。在这里，召回后的诗词id和用户上下文作为模型的输入，得到最终打分，排序后取出top k作为推荐列表返回给用户。

数据格式如下：

|  |  |  |
| --- | --- | --- |
| 列 | 类型 | 描述 |
| poem\_id | categorical | 诗词id |
| age | continuous | 年龄 |
| gender | Categorical | 性别 |
| device\_id | Categorical | 设备类型 |
| weather\_tag | Categorical | 天气标签 |
| wind\_tag | Categorical | 风力标签 |
| temperature\_tag | Categorical | 温度标签 |
| time\_tag | Categorical | 时间标签 |
| season\_tag | Categorical | 季节标签 |
| festival\_tag | Categorical | 节日标签 |
| region\_tag | Categorical | 地点标签 |
| star | categorical | 0,1是否点赞 |

### 4.3 算法评价

由于缺乏训练数据，无法进行准确度等评价，只能采用人主观评价。指定上下文信息后，查看推荐结果。根据推荐结果调整各标签的权重。待后续产生用户数据，即可使用相关数据进行评价。

### 4.4 结果展示

 推荐接口可指定参数或者不指定参数。以下为指定参数的推荐结果

## 结束语

诗词作为一种情感的表达，相似的上下文环境必然更能引发读者的共鸣，基于此，我们充分挖掘了诗词相关数据和用户上下文数据，进行推荐，并取得一定效果。

此处为整体思路介绍。具体代码介绍见github

所有代码放在：<https://github.com/WangShengguang/recommender-system>

访问地址：<http://148.70.6.97/recommend/>

## 参考文献

[1] 陈运文. 用文本挖掘分析了5万首《全唐诗》，竟然发现这些秘密 - 陈运文的文章 - 知乎<https://zhuanlan.zhihu.com/p/45415824> . 2018.10.9/2019.05.04