# 信息检索设计文档

**小组分工：**

崔一鸣： 负责爬虫模块，数据库设计，web后端开发

邹雨婷： 负责web前端开发，搜索自动补全

张 滔： 负责通配符索引的构建

周文涛： 负责检索和网页排名

钱塘文： 负责倒排记录表的构建

### 1. 项目简介

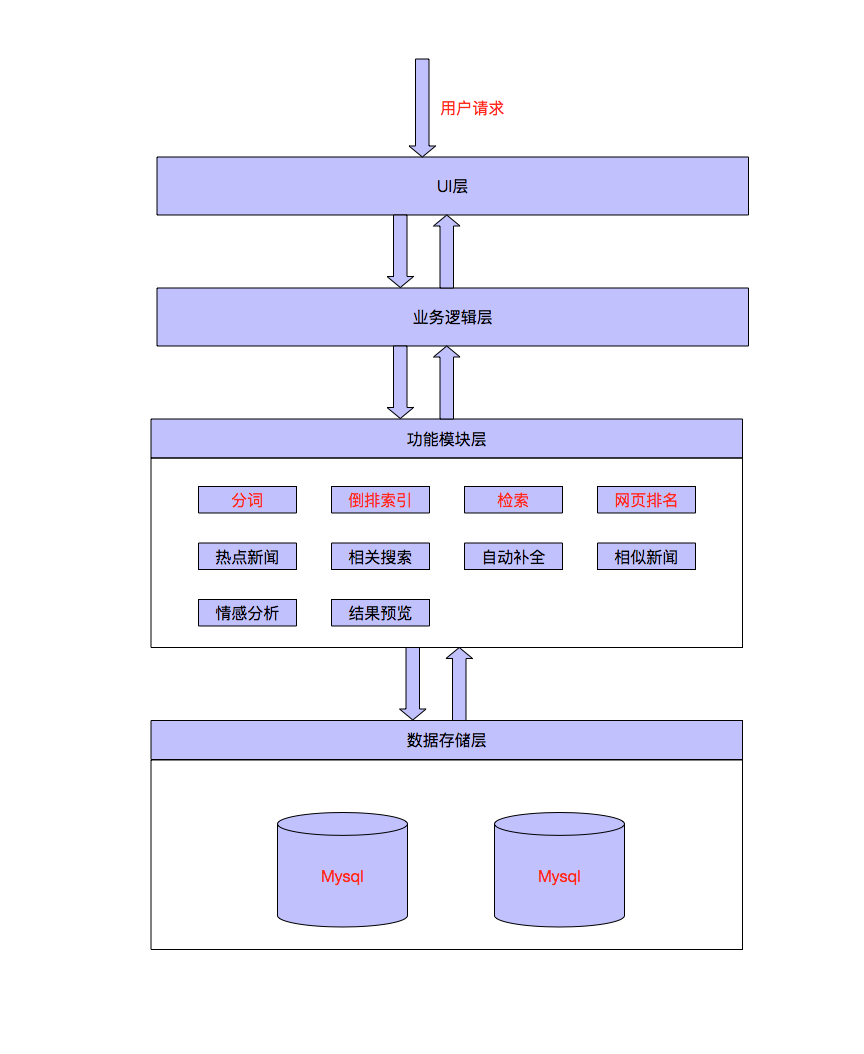
What's News是一个垂直的新闻内容检索网站，收录了网易，搜狐，凤凰，央视等4个新闻网站的10W+条网页。能够对新闻内容，评论进行检索，排名。检索时间小于2秒。同时What's News还提供了热点新闻推荐，搜索自动补全，相关搜索推荐，Snippet生成，搜索结果预览，自定义排序等内容。

What's News后端使用Python的Django框架，核心模块使用java开发。总代码量4k+行。

#### 1.1 搜索引擎地址：(在UCAS网络内访问) <http://123.57.35.217:8888/search>

#### 1.2 源代码地址： <https://github.com/LeoCui/NIR>

### 2. 总体框架



### 3 存储方案设计

#### 3.1 mysql数据库

系统主要采取mysql来进行数据存储方案。mysql数据库中储存的信息包括文档内容信息，搜索历史记录信息，倒排记录表信息等。

注：在操作数据库时，要注意查询的列有没有索引，没有索引的查询会非常慢，但是索引会增加空间。后期视代码逻辑决定要不要给某个列加索引。

建表语句见github上createTable.db文件

## 3.1.1 news\_info表

news\_info表存储的是新闻的基本信息，表中字段包括：

id： 自增id，主键

title: 新闻标题

url: 新闻的原始url

url\_hash: 对url进行hash

pv: 新闻浏览量

category: 新闻分类

is\_handled: 1表示已经被倒排记录表处理，0表示未被处理

comment\_number: 新闻的评论数

publish\_time: 新闻的发布时间

create\_time: 该条记录的创建时间

update\_time: 该条记录的更新时间

extra\_info: 额外选项，方便以后拓展

索引项：pv，url\_hash，comment\_number，publish\_time，这几项建索引是为了最热新闻推荐和rank。

## 3.1.2 content\_info表

content\_info表存储的是新闻的内容部分（不含评论），单独拆开一张表，是因为新闻内容的大小不确定，mysql中变量只能定长。为了节省内存，可能需要把一条新闻内容存在多条记录中。

表中字段包括：

id: 自增id，主键

news\_id: 所在的新闻的id

sequece\_number: 该条记录在该条新闻内容中的序列号。

content: 内容

create\_time: 该条记录中的创建时间

update\_time: 该条记录的更新时间

extra\_info: 额外选项，方便以后拓展

## 3.1.2 comment\_info表

comment\_info表存储的是新闻的评论，之所以要讲新闻和评论分开存储，原因如下：

a. 通常来说，一篇新闻的评论总是在更新（新增评论，删除评论）。但是新闻的内容是不变的，当评论更新时，我们就需要相应地更改倒排记录表，如果新闻内容和评论存储在一起，当评论更新时，我们需要对整个文档修改倒排记录表，分开的话只需要对评论部分修改倒排记录表。

b. 后续可能会有只针对评论信息的检索，那么对底层存储来说就需要将评论和新闻内容分开。

表中字段包括：

id: 自增id，主键

news\_id: 所在的新闻的id

comment\_number: 一条记录里的评论数

content: 评论内容，含有多条评论信息

create\_time: 该条记录中的创建时间

update\_time: 该条记录的更新时间

extra\_info: 额外选项，方便以后拓展

索引：news\_id

注：一条评论包括“日期 时间 用户名 内容”。评论之间用|分开

比如：2017-08-18 13:23:23 开飞机的贝塔 支持威武有希望了||2017-08-13 13：43:34杨总 哈哈

为了防止干扰，爬虫程序在向数据库中插入评论时，需要预处理，保证用户名和内容中不含空格和|

## 3.1.3 search\_history表

主要用于自动补全和相关搜索推荐

表中字段包括：

id：自增id，主键

content: 搜索内容

user\_id: 用户id，暂时不用

user\_ip: 用户ip，暂时不用

create\_time: 该条记录创建时间

update\_time: 该条记录更新时间

extra\_info: 额外选项，方便以后拓展

## 3.1.4 posting\_list表

倒排记录表，主要是为了将内存中建立的倒排记录表持久化存储起来，以便查询时使用。

表中字段如下：

id: 自增id，主键

term: 单词

term\_hash: 该单词对应的hash值，（crc32）

df: 出现该单词的文档的数目

sequence: 该条记录在该term对应的记录的序号

content: 倒排记录表项

create\_time: 该条记录创建的时间

update\_time: 该条记录更新的时间

extra\_info: 额外选项，方便以后拓展

由于term对应的文档数不定，无法确定posting\_list大小，所以可能一个单词对应的文档存储在多条记录中。

posting\_list中一个文档信息的结构如下： 0/1 docID tf <position1,position2>

不同文档之间用|分开其中:

0表示新闻（contetn\_info表），1表示评论（comment\_info表）

docID 对应的news\_id

tf 表示term在docID中出现的次数

<>中保存的是位置信息，用于位置索引

如： 0 34 3 <1,5,8>|1 45 2 <3,4>

注：当content的内容不够一个容纳一个文档的位置信息时，会把该文档的位置信息拆成多份存储。后面用到的时候需要注意。

## 3.1.5 hot\_news表 热点新闻表，主要存储的是最近的最热新闻。

表中字段如下：

id：自增id，主键

news\_id：新闻在news\_info表中的id

rank：该条新闻的热度排名

create\_time：该条记录创建的时间

update\_time：该条记录更新的时间

extra\_info: 额外选项，方便以后拓展

## 3.1.6 dictionary表

词典表，存储的是所有的term

表中字段如下：

id: 自增id，主键

term: 词项

term\_hash: 词项的hash（crc32）

create\_time：该条记录创建的时间

update\_time：该条记录更新的时间

extra\_info: 额外选项，方便以后拓展

## 3.1.7 kgram\_index表

存储的是kgram索引，用来进行通配符查询

表中字段如下：

id: 自增id，主键

kgram: kgram索引

kgram\_hash：kgram的hash值（crc32）

number: 该条记录中content中termId的个数

content: content里面存的是该kgram对应的在dictionary表中的term的id的集合。格式为<termId1, termId2>，一条记录插不下需要拆成多条插入。

create\_time：该条记录创建的时间

update\_time：该条记录更新的时间

extra\_info: 额外选项，方便以后拓展

### 4. 模块设计和实现

#### 4.1 Web服务模块

Web服务负责接收用户的Http请求，然后调用各种服务，进行相应的逻辑处理，最后返回给用户正确的结果。

Web服务分为前端和后端两个部分，前后端是完全分离的。

## 4.1.1 Web后端

web后端使用了Python的Django框架，开发语言为Python，web后端主要负责处理用户的请求，然后得到检索结果，并且将检索结果传递给前端。web后端包含以下几个部分：

##### 4.1.1.1 路由分发

路由分发采用的是django框架的路由分发机制，主要负责**正则匹配**用户的url，然后交给相应的模块去处理。

该部分位于app/search/url.py文件中。

##### 4.1.1.2 逻辑处理部分

这部分主要负责将用户输入的参数封装成Json的格式，然后通过调用各种接口(检索接口，相关搜索接口，历史记录接口，热点新闻接口，snippet生成接口)，接口以Json的格式返回相关的内容，然后将内容返回给前端展示。

该部分位于app/search/views.py

## 4.2.1 web前端

web前端使用了jquery，javascript，ajax，模板等技术，主要负责UI和展示。

这里不再具体阐述。

该部分位于app/search/templates文件夹中。

#### 4.2 爬虫模块

爬虫模块负责爬取网站新闻存入数据库中，包括4个爬虫：网易新闻APP，新浪新闻App，搜狐新闻App，央视新闻App。爬取了10w个网页

爬虫模块使用python编写。

为了防止多次爬取到相同的新闻，在news\_info表中设置了url\_hash这个字段，保证相同url的新闻只被加入数据库中一次。

该部分位于crawler文件夹中

## 4.2.1 抓包

因为移动端的数据格式清晰，所以爬虫选择了移动端新闻App。利用fidder抓包拿到移动端请求url，然后利用requests模拟请求。

## 4.2.2 爬取新闻

这部分负责爬取部分新闻，解析返回的Json字符串，然后按照指定的格式存入数据库中的news\_info，content\_info，comment\_info表中，供建立倒排记录表使用。

为了保持新闻的时效性，爬虫的脚本被加入了系统的定时任务，每天早上8点执行。

## 4.2.3 网页更新

这部分指的是网页内容有变化（评论增加，评论删除）的时候，用来更新数据库（news\_info，comment\_info表）。

这部分和上个模块不同，上个模块指的是爬取了新的网页，本模块指的是已经爬取的网页的内容改变。

当网页内容改变时（比如该网页被删除），除了更改news\_info，comment\_info表之外，还需要删除倒排记录表中含有原有docID的记录。

#### 4.3 倒排索引模块

这部分指的是读取数据库中的新闻，然后建立倒排记录表存入数据库中的posting\_list表中。

该部分用Java编写

## 4.3.1分词segmentation() <不对外接口>

利用HanLP外部工具进行分词，传入参数待分词的句子或文章，返回分词结果arraylist< String>，为后续建立词典表做前期准备.

## 4.3.2 初始化倒排记录表buildPostingList() <对外接口>

读取news\_info中的is\_handled为0的新闻ID，到content\_info表与

comment\_info表中获取相关的新闻正文、评论内容后建立倒排记录表，并按照数据表的设计规则、停用表的规则进行处理、存入posting\_list表中

## 4.3.2 更新倒排记录表updatePostingList() <对外接口>

直接在原有的倒排记录表posting\_list的基础上更新，更新的具体步骤与上面一致

## 4.3.4数据库相关操作，诸如连接、批处理、更新、关闭等mysqlDBInterface.java

将数据库的相关操作进行了泛化封装，使得数据库的操作在前几个功能中变得简单；将数据库的插入、更新操作进行了批处理，加快了数据库执行部分的速度；将数据库的查询操作进行了泛化封装处理，使得前几个功能可以重复使用该功能；

## 4.3.5 配置文件的生成

将诸如数据库的ip地址、端口号、相关数据表名等写入配置文件中，再将配置文件读入程序中进行操作，使得该程序具有一定的泛化性。当诸如数据库的相关信息发生改变时，无需重新打包可执行jar包。

## 4.3.6 停用词表位于配置文件夹中

将停用词表作为配置文件，使得当停用词表更新或者暂停使用时，无需重新打包可执行jar包，一定程度提高了程序的泛化性。

#### 4.4 通配符索引模块

这部分主要负责建立通配符索引并且存入数据库中的kgarm\_index表中，具体地，采用2-gram索引

## 4.4.1 更新词典表putDictionary() <不对外接口>

在初始化倒排记录表、或更新倒排记录表的程序中内部调用该函数，存储倒排记录表posting\_list中已处理新闻中出现的、所有的词项term

## 4.3.2 更新k-gram表handleKGram() <不对外接口>

在更新词典表的程序中内部调用该函数。对词典表中尚未进行k-gram索引处理的词项，建立其k-gram索引，便于进行通配符查询，并按照数据表的设计规则、停用表的规则进行处理、存入kgram\_index表中。

#### 4.4 检索模块

该部分主要接受用户的query，然后返回对应的docID的集合。检索使用hash表的方式。在倒排记录中有字段hash，表示某个term的hash值。

技术：java

该部分需要对外提供一个接口：

query 函数：

输入：json格式，设为input

input['query']: 查询的单词

input['page']；第几页

input['category']: 分类

input['source'] : 来源

input['from']: 开始时间，如2017-12-09 12:00:00

input['to'] : 结束时间

input['sort'] : 排序方式： 0表示按照相关度排序，1表示按照时间排序

输出： json格式，设为output

output['resultCount'] : 所有相关的文档的数量

output[' keywords']： 查询的关键词，是一个list

output['docList']: 相关的文档的信息，是一个list，list成员为doc

doc的结构如下：

doc['id'] : 文档的id

doc['relationship']：文档的相关度，如95.32%

#### 4.5 网页rank模块

这部分主要负责对检索到的文档打分，排序并输出。

技术：java

该部分需要对外提供一个接口：

rank(str，docIDSet，maxNumber): str表示查询的字符串，docIDSet表示相关的文档集合。maxNumber表示返回的文档数目，-1表示不限，返回值是排好序的文档集合。

#### 4.6 其他模块

## 4.6.1 热点新闻推荐

该部分主要负责进行首页的热点新闻推荐。

该模块用Python编写，包括一个定时脚本，每隔

## 4.6.2 搜索自动补全

这部分主要负责根据查询的query给出相应的自动补全提示，调用的是百度的api：http://suggestion.baidu.com/su

主要是前端的一段javascript代码来完成。

## 4.6.3 结果预览

该部分主要用于在检索页面预览内容，而不用点进去。当鼠标悬浮在预览按钮时，会发送一个ajax请求获取该新闻内容，然后以悬浮窗的形式显示出来。

该部分主要由前端的一段javascipt代码完成。

## 4.6.4 获取历史记录

该部分主要负责查询search\_history表，然后获取最新的4条所有记录，返回给前端。

## 4.6.5 snippet生成

该部分负责生成新闻的摘要，因为摘要要尽可能多地包含查询的关键词，所以该部分设计了一个滑动窗口（大小为200个字），遍历该文档，找到一个包含关键词最多的滑动窗口即为snippet，并且将关键字标红显示。

## 4.6.6 相关搜索推荐

该部分主要根据搜索的query获取相关的搜索，该部分调用的是Bing的API:

api.bing.com/osjson.aspx?query=，然后将结果返回给前端

## 4.6.7 自定义筛选

因为数据库中存储了新闻的来源，分类，时间字段，所以该部分支持用户对检索处理的内容进行二次筛选。

当用户点击相关的筛选按钮时，前端会将筛选条件发送给后端的检索模块，检索模块根据筛选条件进行相应的过滤操作，最后返回用户过滤后的结果。

## 4.6.8 自定义排序

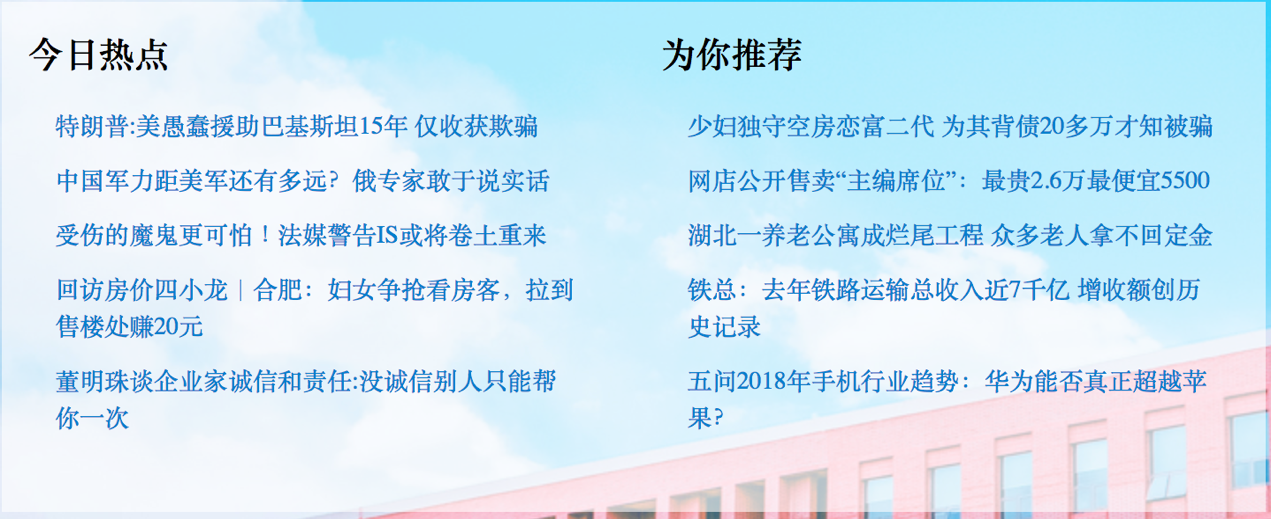
支持按照相关度和发布时间进行排序。

### 5 效果展示

#### 5.1 首页



#### 5.2 热点新闻推荐



#### 5.3 搜索自动补全



#### 5.4 新闻搜索



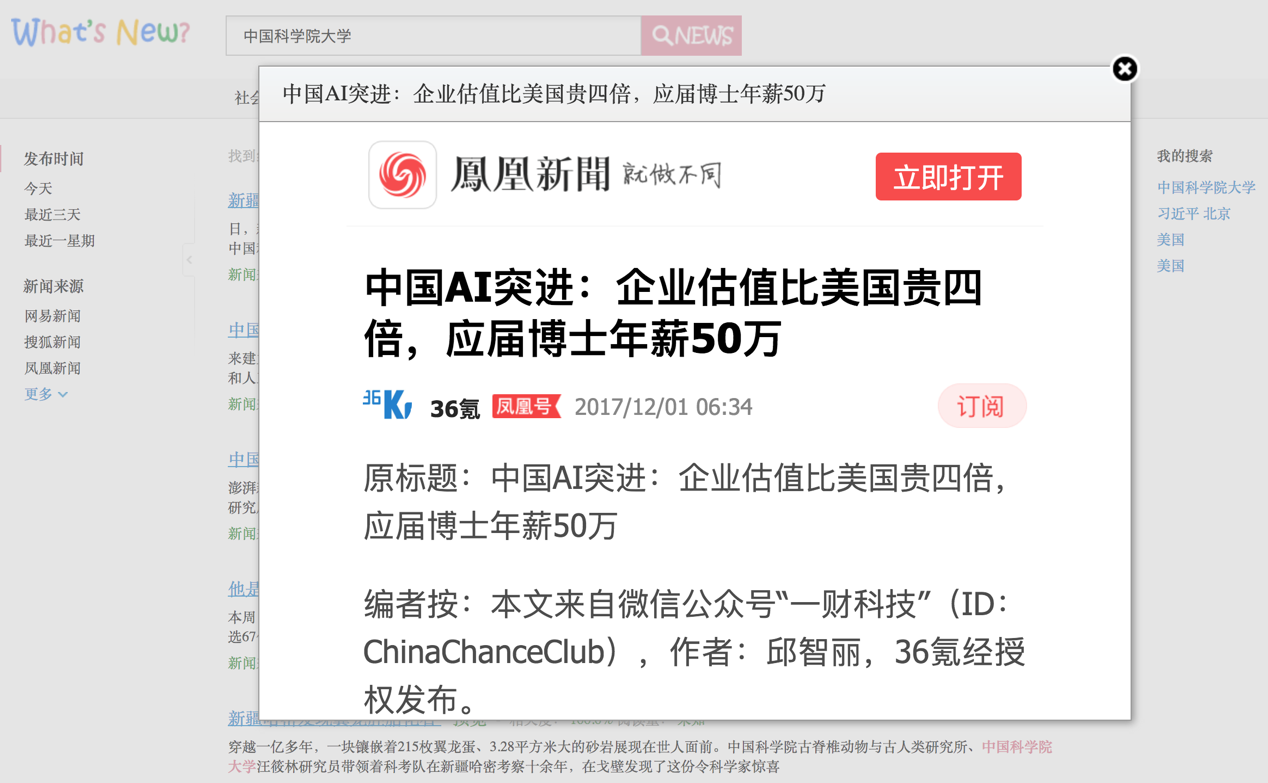
#### 5.5 评论搜索

新闻搜索支持评论搜索，当评论包含相关的内容也会被搜索出来



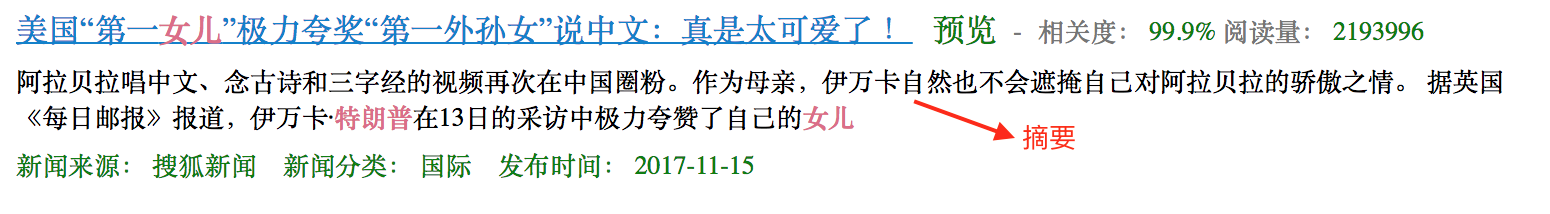
#### 5.6 新闻预览

当鼠标悬浮在新闻标题后面的预览上时，会出现下面的预览浮层。



#### 5.7 snippet生成

在每条新闻的标题的下面有一小段相似新闻的摘要。



#### 5.8 相关搜索推荐

在结果页面的最下面有相关搜索推荐，个数限制在了4个

#### 

#### 5.9 搜索历史推荐

在结果页的右侧有用户的搜索历史的推荐，个数限制在了4个



#### 5.10 自定义筛选

在结果页，用户可以按照分类，来源，时间对检索出来的新闻进行二次筛选。



#### 5.11 自定义排序

在结果页，用户可以按照相关度，发布时间进行排序，默认情况下按照相关度进行排序。



### 6 创新点

数据库用了事务

term\_hash

评论和内容分离

### 7 经验教训