# 目的

在数据引擎中，数据的解析存储以及表示形式决定了数据引擎效率与精准度。但是不仅仅如此，如何与普通用户更好的交互，如何更好的理解用户的自然语言查询也是一大难题。精确理解分析用户的查询，并且快速返回合适的结果必将提升用户的使用体验。语义搜索功能的目的在于准确处理用户的自然语言查询，提升分析准确度，在与知识库，数据服务的对接中起到承上启下的作用。

# 设计概述

为了用户方便的使用该数据引擎寻找特定的信息，数据引擎的使用方法应该更加人性化，并可以为不同领域的非计算机专业人员提供便利。因此，我们希望用户使用日常语言就能够得到他们想要获取的信息。自然语言处理就是研究能实现人与计算机之间用自然语言进行有效通信的各种理论和方法。

该功能通过自然语言处理分析用户输入的查询语句，识别出查询中包含的时间，实体，实体属性，实体领域等，以特定形式返回给Neo4j数据库。并在给用户展示结果后，记录用户的点击选择，生成数据抽取规则，传给数据服务。数据服务能够因此生成符合用户要求的数据。具体流程如下图：

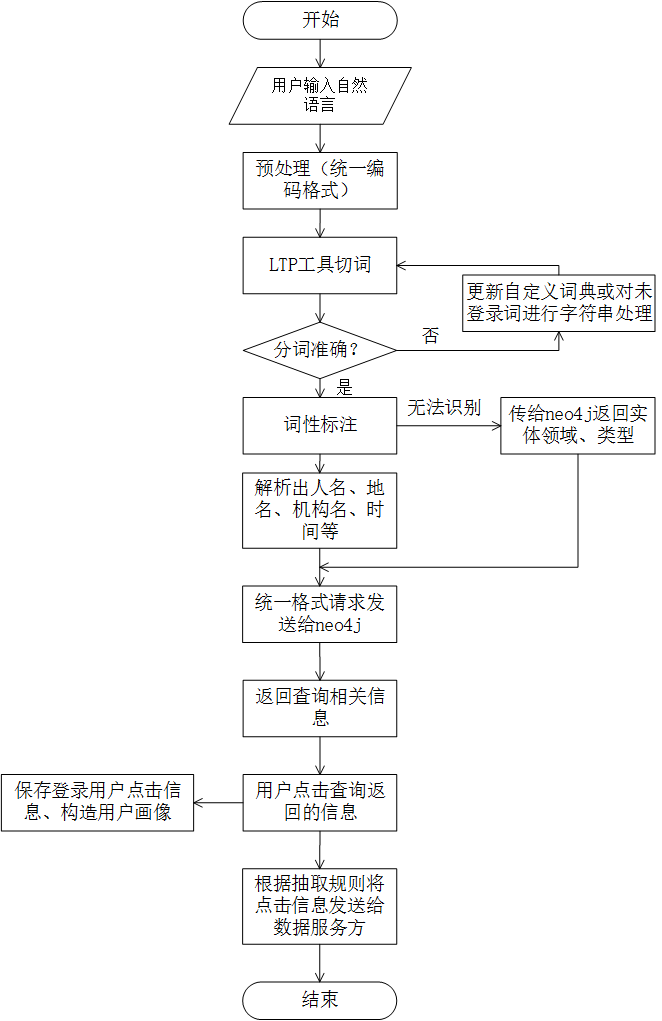


图 2‑1 功能流程

# 业务架构

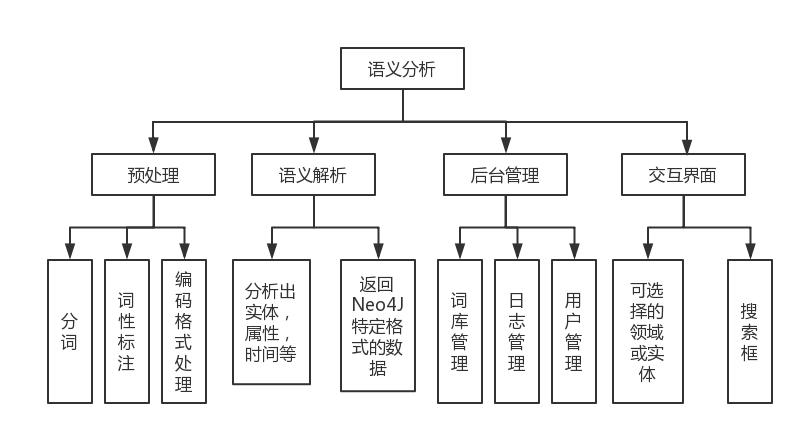


图 3‑1功能模块

整个功能模块如上图所示，它分为四个模块，分别为预处理，语义解析，后台管理以及交互界面。预处理主要针对用户输入的查询语言做处理，排除一些错别字，编码格式的错误并且对其分词，词性标注，方便从语料库中获得已有的信息。语义解析是该功能的难点，它的目的在于获得用户查询语言中包含的时间，实体，实体属性，实体类别等信息，并以json格式返回给Neo4j，能够方便数据库返回符合用户想要的结果。后台管理主要有日志管理，词库管理和用户管理三个功能。词库管理完善了平时无法分词的词语。日志管理保存了用户的输入，系统的报错等内容方便管理员维护。在用户管理中，除了一般需要保存的用户账号，密码等安全信息外，我们还保存用户的搜索历史相关领域和关注领域，借此建立简单的用户画像，向用户推荐数据，提高数据的命中率。在界面设计中，我们争取做到简洁易懂，包括搜索界面，返回的结果界面以及用户注册登录界面。

# 详细设计

## 4.1 内部模块

### 预处理模块

该模块采用了哈工大的LTP工具，统一采用UTF-8模式，将用户的输入进行分词、词性标注。我们初步得到表示时间，地点以及实体的词。

LTP分词准确率

|  | P | R | F |
| --- | --- | --- | --- |
| 开发集 | 0.973152 | 0.972430 | 0.972791 |
| 测试集 | 0.972316 | 0.970354 | 0.972433 |

LTP词性标注准确率：

|  |  |
| --- | --- |
|  | P |
| 开发集 | 0.979621 |
| 测试集 | 0.978337 |

**构建工程：**

* 导入ltp4j.jar
* windows下将libs文件夹中生成的所有动态库、以及原ltp lib文件夹下的splitsnt、segmentor、postagger、ner、parser、srl 6个动态库拷贝到项目根目录。
* linux下export LD\_LIBRARY\_PATH=#jni动态库路径#

**分词接口：**

Int Segmentor.create（String path, String lexicon\_path）

Int Segmentor.create（String path）

功能：

读取模型文件，初始化分词器。

参数名 | 参数描述

String path | 指定模型文件的路径

String lexicon\_path | 指定外部词典路径。如果lexicon\_path为NULL，则不加载外部词典

返回值：

成功加载模型返回1，否则返回-1。

int Segment(String string, java.util.List< String > words)

功能：

调用分词接口。

参数：

参数名 | 参数描述

String | 待分词句子

java.util.List< String > words | 结果分词序列

返回值：

返回结果中词的个数。

**词性标注接口:**

int Postagger.create(String path)

功能：读取模型文件，初始化词性标注器

参数：

参数名 | 参数描述

String path | 词性标注模型路径

返回值：

成功加载模型返回1，否则返回-1。

int Postag(java.util.List< String > words, java.util.List< String > tags)

功能：

调用词性标注接口

参数：

参数名 | 参数描述

java.util.List< String > words | 待标注的词序列

java.util.List< String > tags | 词性标注结果，序列中的第i个元素是第i个词的词性

返回值：

返回结果中词的个数

最后系统能够得到标记为时间（t），地点（ns）， 特有名词（nz），英文缩写（ws）等词性的词。接下来就需要进行语义解析得到更加精确的结果。

### 语义解析

* 1. 时间解析

如果用户输入表示时间的词，其形式可能是多种多样的，比如：“昨天”，“最近三个月”，“前5天”或者就是简单的日期：“2017-8-16”等，但是发送给后台的数据格式必须统一，所以在此需要进行一些处理和判断，目前的方法就是考虑各种情况，增加多种判断语句，并进行不断测试。

* 1. 实体解析

在分词和词性标注之后，我们会得到词性为nz（特有名词）或者ws（外文词语）的词，这类词一般就是我们需要查询的实体，比如商店，品牌等。具体流程图如下：

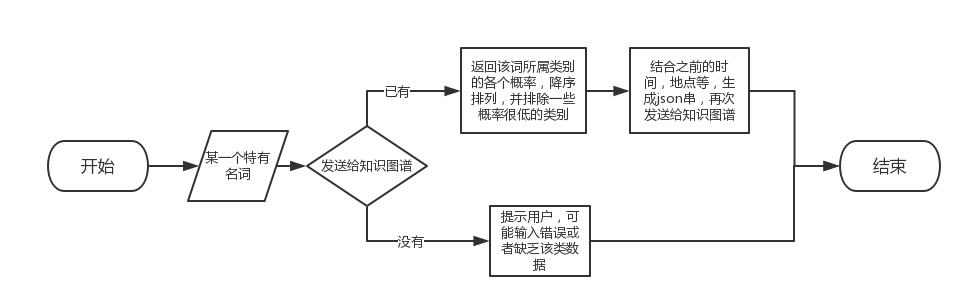


图 4‑1 实体解析流程图

### 日志管理

日志管理采用logback。Logback是由log4j创始人设计的另一个开源日志组件，性能比log4j更快，初始内存加载也小了很多。logback当前分成三个模块：logback-core,logback- classic和logback-access。logback-core是其它两个模块的基础模块。logback-classic是log4j的一个 改良版本。

基本步骤：

1. 添加Maven依赖
2. 配置logback.xml
3. 在需要记录日志的程序中引用Logback：

static final Logger LOG = LoggerFactory.getLogger(MyClassName.class);

### 前端界面展示

参考维基百科以及Babelnet语义网络的界面设计。争取首页简洁明了。前端框架则采用Vue.js。并借助Element UI 组件库。Element UI 是一套采用 Vue 2.0 作为基础框架实现的组件库，能够快速地搭建网站。

在接收到知识图谱返回的结果

……

### 用户管理

在用户管理中，除了普通用户，管理员的注册登陆以外，我们还记录用户的搜索历史，关注领域，关注实体等，建立简略的用户画像系统。借此可以在用户搜索的同时进行数据的推荐，提高数据的命中率。数据库采用MySQL数据库。

**用户管理模块概述：**

该模块分为普通用户和管理员两种角色。普通用户可以通过注册来获得登录账户和密码，管理员直接登录。普通用户登陆后包括该用户的搜索历史相关领域，关注领域、实体以及修改密码，如表4-1，管理员能查看用户信息列表，统计每个领域关注的人数。

**用户管理模块技术分析：**

管理员账户密码直接赋予，用户注册时设置密码。密码在前台与后台数据库均不可见，采用MD5加密算法。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | 账户名 | 密码 | 搜索历史相关领域编号 | 关注领域编号 |
|  |  |  |  |  |

**表4-1 用户信息表**

|  |  |
| --- | --- |
| 领域编号 | 领域名称 |
|  |  |

**用户画像：**

用户画像的核心工作是为用户打标签，打标签的重要目的之一是为了让人能够理解并且方便计算机处理。一个标签通常是人为规定的高度精炼的特征标识，如年龄段标签：25~35岁，地域标签：北京，标签呈现出两个重要特征：语义化，人能很方便地理解每个标签含义。这也使得用户画像模型具备实际意义。能够较好的满足业务需求。如，判断用户偏好。短文本，每个标签通常只表示一种含义，标签本身无需再做过多文本分析等预处理工作，这为利用机器提取标准化信息提供了便利。

在该系统中，我们暂时为用户打上两个标签：搜索历史的相关领域和用户的关注领域。用户搜索历史涉及的领域次数以及关注领域二者通过权重求值后可以得到用户在不同领域的值，这样用户在搜索的时候，可以根据用户的权值来对搜索结果排序推荐。



图 4‑2用户画像

## 4.2 外部接口模块

### 知识库接口

在知识库接口交互中，我们可能需要多次交互。

第一次是在语义解析中，当我们解析出特有名词时，以特定格式发送给知识库，知识库会返回该名词所属类别的字符串”S1:P1，S2:P2,…, Si:Pi”(Si为类别名称，Pi 为属于该类的概率)



在得到字符串之后，我们在结合之前得到的时间，地点等，生成json，再次发送给知识库。知识库通过该json查询得到的结果返回给系统，系统将其进行展示。

### 数据服务接口

系统先记录用户的点击选择，生成数据抽取规则然后通过RabbitMQ发送给数据服务层。

## 4.3 开发环境

开发框架：SpringMVC

数据库：MySQL

服务器：Tomcat 7.0

SDK: JDK 1.7

部署系统：CentOS 7.0