# 第3章 需求分析及设计

## 3.1 系统需求分析

### 3.1.1 用户需求

中文文本审校系统目前只设计了一个用户角色，就是系统的会员用户。用户需要先注册成该系统的会员才能使用该系统的功能。该用户可以自己维护自己的个人信息。同时该用户还能够使用在线和离线两种方式使用该系统对文本进行审校。

### 3.1.2 业务需求

该系统主要包含两个业务，即在线审校和离线审校。以下就分别介绍这两个业务。

1 在线审校

用户可以使用在线审校校对小段文本。在线审校的优点是审校速度快，短时间内就能够返回审校结果，其缺点是在线审校只能校对短文本，有一定的字数限制。

2 离线审校

离线审校则与在线审校不同，离线审校可以允许用户上传Word文档，后台再对该文本进行校对，校对完成之后会在用户界面提示，最后用户再将校对好的Word文件下载下来。

### 3.1.3 功能需求

由用户需求和业务需求的分析可以知道该系统存在一个用户角色，即该系统的会员，用户首先需要注册成为该系统的会员。注册成功后用户可以修改自己的个人信息，同时还可以使用离线审校和在线审校的两种系统所提供的服务，离线审校需要用户先上传word文件然后才能进行审校，审校完成后还需要下载审校结果文件，审校结果文件仍然是word文件。根据这些信息可以得到该系统的用例图，如图3-1所示。以下就分别对这几个用例做出描述。

1 注册

（1）说明

若用户想要使用本系统，首先需要进行个人信息注册，只有注册为本系统的会员才能使用该系统。

（2）事件流

（a）用户通过浏览器进入该系统登录界面，选择注册，系统转到注册页面；

（b）用户填写用户名、密码以及确认密码，用户名为长度为6-20的字母、数字或下划线组成；

（c）系统验证用户名是否有效以及是否重复，同时还需要验证两次密码是否一致，若验证不通过则给出相关提示，直至用户名及密码通过检验；

（d）系统自动生成用户ID，并提示注册已成功，跳转至系统登录页面。

（3）前置条件

用户进入系统注册页面，并申请注册。

（4）后置条件

注册成功，系统添加用户个人信息至数据库。



图3-1 系统用例图

2 登录

（1）说明

会员需要提供正确的用户名及密码，才能登录该系统，使用该系统所提供的服务。

（2）事件流

（a）会员输入该系统的网址，进入登入页面；

（b）会员输入正确的用户名及密码；

（c）系统验证用户名及密码是否正确，若不正确则给出提示，若用户名密码都正确则跳转至该系统首页。

（3）前置条件

用户必须已经注册成为该系统的会员。

（4）后置条件

会员成功登录该系统。

3 个人信息维护

（1）说明

系统会员在完成登录后可以根据自己的需要修改个人信息。

（2）事件流

（a）系统会员在完成登录后，选择个人信息修改，进入个人信息维护界面；

（b）会员填写需要修改的信息，然后点击完成；

（c）系统验证会员所提供信息的有效性，若均有效则提示信息修改成功。

（3）前置条件

会员必须登录该系统。

（4）后置条件

会员信息修改成功。

4 在线审校

（1）说明

该系统的会员可以使用本系统所提供的在线审校功能来校对短文本。

（2）事件流

（a）系统会员在完成登录后，选择在线审校，进入在线审校界面；

（b）用户输入带审校文本，点击审校按钮；

（c）系统后台对用户输入的文本进行审校，审校完成后返回审校结果；

（d）在线审校界面完成对审校结果的显示。

（3）前置条件

会员必须登录该系统。

（4）后置条件

无

5 离线审校

（1）说明

该系统的会员可以上传word文件，并使用本系统的离线审校功能来完成对已上传文件的审校。

（2）事件流

（a）系统会员在完成登录后，选择离线审校，进入离线审校界面；

（b）会员从本地将待审校的word文件上传至服务器中；

（c）word文件上传成功之后，会员发出离线审校请求；

（d）服务器接收到离线审校请求，调用服务器端审校程序，对上传到服务器上的文件进行审校，并将审校结果写入一个新的word文件中。

（e）前台给出审校成功的提示，用户选择审校好的文件进行下载。

（3）前置条件

会员必须登录该系统。

（4）后置条件

无。

## 3.2 系统设计

经过3.1的系统需求得出了该系统的主要的功能，本节将详细的介绍该系统的设计。

### 3.2.1 系统体系结构设计

中文文本自动校对系统是基于B/S的网络服务系统，采用的是Spring MVC的三层Web架构。所谓的MVC架构就是将视图层（View）、控制层（Controller）和模型层（Model）分开，这样可以极大的降低程序的耦合程度，便于以后对系统的维护和升级。该系统的整体架构图如图3-2所示。其中该系统的View层为用户与该系统交互的接口，用户所能看到的所有界面都可以称为视图层，该层主要用Jsp实现，而且还会用到一些JavaScript、CSS和JSTL标签等这些前端常用的技术。该系统的前台主要包含在线审校、离线审校和个人信息维护三部分功能。



图3-2 系统架构图

因为本系统采用的是Spring MVC，所以该系统的控制层采用的还是原生的Java编写的。控制层主要负责接收前台的action请求，并确定调用哪部分后台功能来处理前台的action请求。

模型层主要包括业务层和数据持久层（Dao），业务层主要负责处理业务逻辑，而数据持久层主要负责对数据库进行数据的存取操作，这样就进一步降低了系统的耦合性。该系统后台的业务主要包括两个部分功能：审校服务功能和个人信息维护功能。其中审校服务主要完成对中文文本进行校对的工作，而个人信息维护则是完成对个人信息表的查询和修改。

由于在离线审校的时候需要在服务器端使用审校服务校对用户上传的文件，所以本文将审校服务独立成一个API。这样既能在在线审校时调用审校服务，同时又方便了在后台启动离线审校服务。

### 3.2.2 审校服务功能设计

该系统的审校服务主要针对的是现有的中文网络文本常见的中文词语、汉语拼音、数字号码、以及标点符号等常见的错误进行审校。审校服务具体的功能框架图如图3-3所示。其中每块具体所需要审校的内容如下



图3-3 审校服务功能

（1）词语审校

包括由于音近、形近、义近等因素导致的人名、地名、机构名等实体名称及敏感词语的审校等，如将“王歧山”校正为“王岐山”、将“荷泽”校正为“菏泽”、将“大义禀然”校正为“大义凛然”及将“寒喧”校正为“寒暄”等。

（2）拼音审校

包括字符审校与大小写审校等，如将“Màodùn（冒顿）”校正为“Mòdú”及将“Zhāngyáng Wénān（张杨文安）”校正为“Zhāng-Yáng Wénān”等。

（3）数字审校

数字审校，包括日期、年龄、号码、专用数字等数字的审校，主要通过设定正则表达式形式的审校规则实现，如标注车牌号、手机号及专用数字中的错误等。

（4）标点符号审校

标点符号审校，包括标点符号的误用审校、冗余审校及遗漏审校等，如将误用的问号校正为逗号、剔除冗余的引号及添加遗漏的句号等。

## 3.3 本章小结

本章首先从用户需求、业务需求和功能需求三个角度分析了该系统的需求，紧接着从MVC三层架构分析了该系统的体系结构设计，最后对审校服务的内部功能进行了设计。

# 第4章 中文文本审校功能

对文本进行审校是中文文本自动校对系统的主要核心功能。具体包括词语审校，标点符号审校，数字审校和拼音审校。本章将详细阐述这几个功能的具体实现方法。

## 4.1 词语审校

词语审校实际上是指错别字审校，即校正词语中的一个或多个错误用字。在中文文本，尤其是网络文本中，错别字极为常见。在这里，将词语审校划分为人名、地名、机构名等实体名称和常用词的审校。词语审校的主要方式是采用错别字对照表的方式进行。现今的大多数文本审校软件，包括百度词典、有道词典等在线中文辞典及百度中文输入法、Sogou中文输入法等均提供有海量的词库实现对词语的审校，词语审校的主要依据就是其所提供的词汇库。黑马校对V18更是提供有5000万专业词汇，Sogou中文输入法的中文词汇也已超过4000万。海量的词汇是中文文本审校的基石，为了更有效的提供中文审校服务，必须构建足够数量的中文词汇库。在词语审校方面，区别于通常的专业词语审校，可以将词语分为人名、地名、成语、机构名、敏感词语及其它词语等。通过构建词语审校表或审校规则实现不同类别词语的审校。下面分别介绍不同类别词语的审校方案。

### 4.1.1 实体名称审校

实体名称审校其实就是判断文本中的实体名称与知识库中的实体是否一致的过程，若不一致，则用知识库中的实体名称对文本中的实体名称进行纠正。实体名称审校具体分为以下几个部分。

1、知识库的构建

实体（Entity）是自身客观存在并具有唯一性的具体事物。实体有别于本体，本体是现实生活中某一类实体的总称。本文为了审校实体名称首先需要构建一个包含大量实体的知识库。本文知识库的构建过程如下：

（1）爬取网页文本

构建实体知识库需要有大量的实体信息，而要获取这些大量的实体信息则需要从互联网上爬取大量的网页文本。本文采用的数据源为百度百科。采用百度百科作为数据源原因是：百度百科涵盖了大量的实体信息，而且覆盖面极其广泛；百度百科实体信息更新频率高，百度百科允许所有用户自己创建和编辑词条，这就使得更多的用户可以参与到创建和编辑百度百科的词条中来。所以通过爬取百度百科可以获取到更多质量更高的实体信息。

百度百科的大多数信息都是以HTML网页的形式展现给用户的。所以要获取到百度百科中的实体信息，首先要做的事就是将百度百科的HTML网页文本给爬取下来。如图4-1为四大天王之一刘德华的百度百科词条页面及其对应的HTML源代码。所以要抽取该实体的信息，首先需要把其对应的网页HTML文本爬取下来。本文总共从百度百科网页上爬取了1440000个HTML网页页面。其中部分列表如图4-2所示。



图4-1 百度百科词条及对应HTML源码

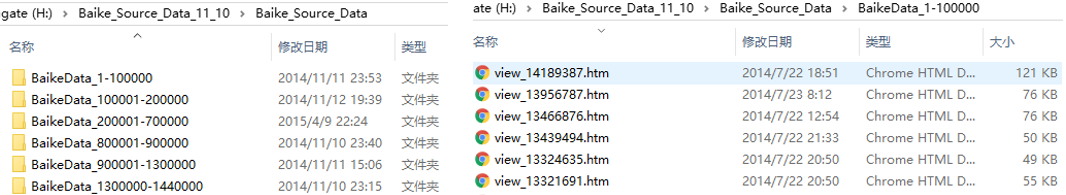


图4-2 百度百科部分网页页面列表

（2）抽取并保存网页文本中的实体信息

在完成网页文本爬取及保存后，接下来要做的工作就是抽取网页HTML文本中的信息。由图4-1可以看出，HTML源码都具有一定的规律性，每一段信息的div都有其唯一的class相对应，就如图中所示第一段的概要的对应的div的class为“lemma-Summary”，而基本信息对应div的class为“basic-info”。在明确每段内容所对应的class之后只需要正则表达式就很容易能从中把要抽取的内容抽取出来。本文目前只需要实体的基本信息，所以在抽取的时候只是抽取了实体的基本信息。首先需要获取文件夹中的网页文本的数目，即遍历每一个HTML文件，再则是将文件读入到内存中，其次则是利用正则表达式Regex=”basic-Info-block” 抽取每个网页中的基本信息。在完成抽取后，则是将实体的基本信息存放到MongoDB数据库中。具体的抽取流程如图4-3所示。



图4-3 百科页面抽取流程图

2、实体识别

本文的实体识别采用的是西南交通大学分词和CRF模型相结合的方法。实体识别的详细流程如图4-4所示。

（1）利用分词的词性进行实体识别

分词是中文自然语言处理的一个重要的课题，大多数的中文自然语言应用都必须以中文分词为基础。本文使用西南交通大学分词系统对中文文本进行分词。因为西南交大分词系统分完词后会带有词性标注，所以本文利用分词后文本中的词性标注来做部分命名实体识别，西南交大分词系统具体的词性和词性说明如表4-1所示。具体步骤分为两步：首先利用分词系统对文本进行分词；其次，查询分词后的文本，若文本中出现词性为nr、ns、nt的词，则将其词性改为entity。因为本文只考虑对人名、地名和机构名的识别，所以本文进行实体识别时，也只取nr、ns、nt三种词性。

表4-1 西南交大分词系统词性和词性说明表

|  |  |  |  |
| --- | --- | --- | --- |
| 词性 | 词性说明 | 词性 | 词性说明 |
| nr | 人名 | nt | 机构团体名 |
| ns | 地名 | nh | 医药疾病等健康相关名词 |
| nb | 生物名 | nf | 食品 |
| g | 学术词汇 | w | 标点符号 |
| u | 助词 | m | 数量词 |



图4-4 实体识别流程

（2）利用CRF进行实体识别

中文文本在经过分词之后有些命名实体会被拆分为多个词语，例如“成都汇康中西医结合医院”分词后的结果为“成都/ns 汇/ng 康/nz 中西医结合医院/nth”，这样就无法准确的通过分词的词性标注来识别出正确的实体名称。而CRF则具有组块的功能，可将被拆开的实体名称组合成一个完整的命名实体。

使用CRF做命名实体识别首先要做的事就是确定特征模板。特征模板一般采用当前词与前后个词及其标记进行组合，即以当前词与前后*n*个词及其标记作为观测窗口。由于当窗口过大时算法的执行效率会过低，而且准确率也不一定会很高，而当窗口过小时所涵盖的信息又太少，不足以完成命名实体识别的任务，因此，在一般情况下*n*的取值范围为2~3。而本文的*n*取值为2，具体的特征模板如表4-2所示。

表4-2 CRF特征模板

|  |  |  |  |
| --- | --- | --- | --- |
| ID | 特征模版 | ID | 特征模版 |
| 1 | U01:%x[-2,0] | 8 | U08:%x[-2,0]/%x[-1,0] |
| 2 | U02:%x[-1,0] | 9 | U09:%x[-1,0]/%x[0,0] |
| 3 | U03:%x[0,0] | 10 | U10:%x[0,0]/%x[1,0] |
| 4 | U04:%x[1,0] | 11 | U11:%x[1,0]/%x[2,0] |
| 5 | U05:%x[2,0] | 12 | U12:%x[-2,0]/%x[-1,0]/%x[0,0] |
| 6 | U06:%x[-1,0]/%x[0,0] | 13 | U13:%x[-1,0]/%x[0,0]/%x[1,0] |
| 7 | U07:%x[0,0]/%x[1,0] | 14 | U14:%x[0,0]/%x[1,0]/%x[2,0] |

基于CRF的命名实体识别是一种有监督的识别方法，因此需要用大量的已标注语料来对CRF模型进行参数训练。本文采用的训练语料为NLPCC2016问答评测任务所提供的训练数据，该数据集总共有14609条句子，本文取了其中的10000条作为训练数据，并对训练数据进行了标注，其中部分标注好的训练数据如图4-5所示，其中用中括号（[]）括起来的就是标注好的实体。

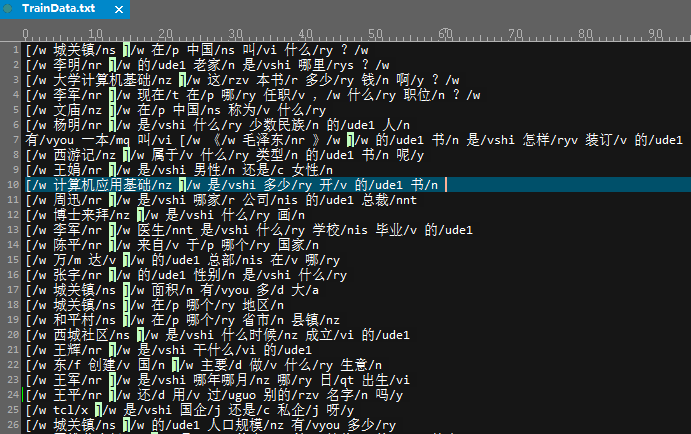


图4-5 CRF命名实体识别部分标注数据

3、实体链接

命名实体链接是指将文本中已经识别出的命名实体链接到知识库中的一个具体真实实体的过程。实体链接是目前自然语言处理领域的一种常用技术。本文的实体链接主要分为以下几个步骤。

（1）同义词表构建

因为本文是文本校对系统，所以在文本中出现的实体名称有可能会是错误的名称，而直接用错误的名称查知识库有可能查不到相应的实体，而且在汉语文本中实体的名称可以是灵活多变的，因为很多名称都存在简称或别名等其他称呼（如科比布莱恩特，别名或简称可为科比、小飞侠、黑曼巴等等），而知识库中存的则是标准的全名，所以当直接用别名或简称来查知识库时也查找不到对应的实体，这就需要构建同义词表。

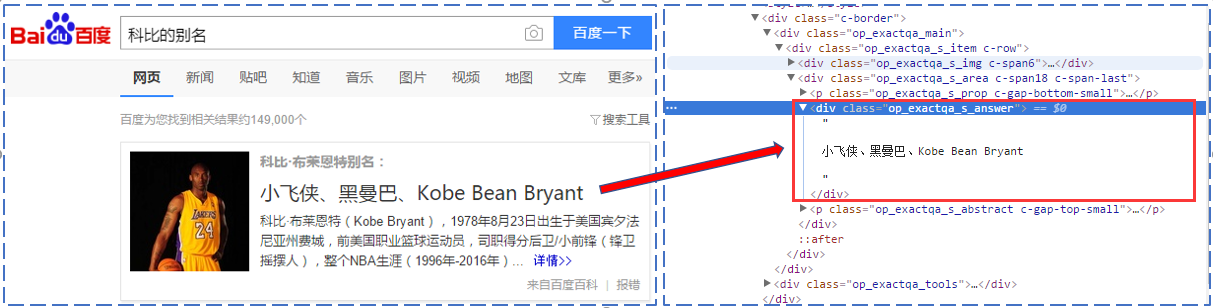


图4-6 百度同义词示例



图4-7 百度百科词条基本信息示例

百度在百度一个人的别名的时候多数情况下会自动给返回其抽取好的别名，如图4-6所示，为百度“科比的别名”后的结果，百度会自动给出其抽取好的别名。本文的同义词表就是借助百度的这个功能来构建的。从图中还可以看出其别名是放在一个class为“op\_exactqa\_s\_answer”的div中。所以在抽取时只需要抽取该div中的内容即可。但也有些情况在百度一个实体名称的别名时候百度并不会直接给出答案，所以这时候需要获取该实体的百度百科词条页面中的基本信息，因为很多实体的百科词条页面中会包含该实体的别名，示例如图4-7所示。这时候只需要抽取该实体百度百科词条页面中的别名即可。具体同义词表构建的算法4-1所示。同义词表达格式为“实体名称 别名1、别名2、… 、别名n”。以歌星陈奕迅为例，其同义词表的格式为：“陈奕迅 医生、吹神、陈小胖、所长”。

|  |
| --- |
| **算法4-1** ：同义词表构建算法 |
| **输入**：实体的名称entityName  **输出**：抽取出的同义词表nickNames  **begin**  1) **DownLoad** content←download(entityName+"的别名") /\*利用百度引擎搜索实体名称的别名并保存其结果\*/  2) **Intialize** XPath Engine htmlCleaner /\*初始化XPath处理引擎\*/  3) **compute** rootNode**←**htmlCleaner(*content*) /\* 将页面源码*content*转化为XPath表达式树\*/  4) **nickNames**←rootNode.evaluateXPath(".//div[@class='op\_exactqa\_s\_answer']") /\*利用XPath表达式抽取其中class为op\_exactqa\_s\_answer中的内容\*/  5) **if** nickNames为空 /\*若不能直接通过百度别名抽取出其别名\*/  6) **entityUrl**←getUrl(entityName) /\*利用百度引擎搜索实体名称并抽取其百度百科的url\*/  7) **if** entityUrl不为空  8) **DownLoad** baikeContent←download(entityUrl) /\*保存实体百科页面\*/  9) **compute** rootNode**←**htmlCleaner(*content*) /\* 将页面源码*content*转化为XPath表达式树\*/  10) **infobox←**rootNode .evaluateXPath(".//div[@class = 'basic-info']"); /\*利用XPath表达式抽取百科页面中的基本信息\*/  10) **nickNames←**getNickName(infobox) /\*抽取infobox中的别名\*/  10）  **end if**  11) **end if**  18) **return** nickNames  **End** |

（2）实体链接

有了同义词表后，接下来的工作就是进行实体链接。本文的实体链接分为两个步骤，首先用实体名称直接查找，若用实体名称无法找到，则通过匹配同义词表查找。直接查找比较简单，就是通过查询语句用实体名称直接查询知识库，这里就不做详细的介绍了。但是对于那些可能有错词或者是别名的实体名称，而知识库中存的都是标准的名称，这样就无法直接查找知识库找出想要的实体，所以就必须通过相应的办法找出完整准确的实体名称，再用正确的实体名称去查询知识库找出相应的实体。本文采用的就是同义词表匹配的方法。

同义词表匹配的方法首先是遍历同义词表，看是否存在与实体名称*entityName*长度一致的词*nickName*，若存在计算*nickName*与*entityName*的拼音相似度*spellSimilarity*。本文用得是最长公共子序列去计算两个词的拼音相似度具体计算方法如下：设entityName的拼音长度为*n*，nickName的拼音长度为*m*，entityName的拼音与nickName的拼音中的最长公共子序列长度为*k*，则*spellSimilarity*的计算公式如下：

 (4-1)

拼音相似度主要解决的是由于用户使用拼音输入法导致的实体名称错误，而大多数人在使用拼音输入法将实体名称打错时，错误的实体名称与正确的实体名称之间的拼音差别一般不会太大，甚至很多情况下是一致的。所以在使用拼音输入法来匹配正确的词的时候相似度的阈值会比较大，本文经过实验调参将拼音相似度的阈值设置为0.95，当则用该*nickName*对应的标准的实体名称去查知识库。

实体名称出现错误一部分原因是因为拼音相似或相近导致的，还有一部分可能是由于字形相似或相近导致的。而字形相似或相近的实体名称，其拼音可能差距会比较大，因而用拼音相似度的算法无法找出正确的实体名称。这就需要计算另外一种相似度。受拼音相似度启发，本文采用了一种汉字的笔画相似度来解决此类由于字形相近而导致的错误。当时，就计算entityName和nickName二者的笔画相似度*strokesSimilarity*。笔画相似度同样采用的是最长公共子序列来计算。设entityName笔画的长度为n，nickName的笔画长度为m，二者笔画的最长公共子序列的长度为k，则*strokesSimilarity*的计算公式如下：

 (4-2)

两个词的笔画相似度的差异性会比拼音相似度要大一些，所以笔画相似度的阈值的取值会比拼音相似度的阈值要小一些。本文经实验验证将笔画相似度的阈值设置为0.86。当时，则用该nickName对应的标准的实体名称去查知识库。

4、实体名称审校

实体名称审校其实就是比较文本中的实体名称与知识库中的实体名称是否一致的过程。在完成实体链接之后如果知识库中存在该名称的实体，若链接成功则会返回一个实体的基本信息，若链接不成功则表示该实体在知识库中不存在，返回空值。若知识库中存在与文本中实体名称相同或者相似的实体，则比较返回实体的实体名称与文本中的实体名称是否一致，若一致，则表示该处实体名称没有错误，不予审校，否则用知识库中的实体名称来纠正文本中的实体名称。若知识库中不存在该实体名称的实体，跳过该处实体名称。

### 4.1.2 常用词语审校

实体名称审校只是整个词语审校中的一部分，文本中有很多的错误还是集中在普通词汇上面。所以常用的普通词汇的审校也是词语审校的一个重要的内容。词语审校主要分为以下几个部分。

1、词库的构建

普通词语审校主要采用的是与词库中正确的词进行比较来实现，所以词库是普通词语审校的基础。本文采用的词库来自搜狗细胞词库。

2、字典树构建

普通词语需要大量的词库，而如何高效地从大量的词库中查找出特定词语就很重要。本文采用了字典树的方式来加载词库，以下就是字典树的介绍及构建过程。

（1）字典树介绍

在计算机科学中，字典树（Trie树）也被称作单词查找树，它是一种哈希树变种的搜索树。字典树主要应用于统计、排序和保存大量的字符串（但并不仅仅限制于字符串）。字典树的键值通常就是字符串，与二叉搜索树不同的是字典树上每个节点的键值都是依赖于其本身的，并不依赖于其他键值。节点的所有后代都具有与该节点相关联的字符串的公共前缀，而且字典树的根节点为空节点。字典树的值不一定与每一个节点相关联，相反字典树的值一般倾向于仅与叶子节点关联。在图4-8的字典树的例子可以看出，每个非叶子节点上只存储key值（单词公共前缀），并不存储value值（单词的词频）。每个单词的词频只在叶子节点存储。



图4-8 字典树示例

（2）字典树构建

字典树构建首先需要构建出字典树的节点类。由上文介绍可知，字典树的非叶子节点与叶子节点是有区别的，有些值只在叶子节点存储，而非叶子节点则不需要存储。所以本文需要构建两个类来存放字典树的节点，一个用来实例化非叶子节点，另一个类用来实例化叶子节点。两个类具体的代码结构如代码4-1和代码4-2所示。非叶子节点只包含一个Map，该Map的key值为一个汉字，表示一个词的第n个字，而其value值则是一个TreeNode的对象。而叶子节点则是继承非叶子节点类，在非叶子节点的基础上增加了词和词频这两个属性。

|  |
| --- |
| **代码4-1** 非叶子节点类 |
| **class** TreeNode {  Map<Character, TreeNode> subNodes; // 子节点  public TreeNode() { //创建新的节点  if (subNodes == null)  subNodes = new HashMap<Character, TreeNode>();  }  public void insert(char k, TreeNode sub) { // 插入节点  if (subNodes == null)  subNodes = new HashMap<Character, TreeNode>();  subNodes.put(k, sub);  }  public TreeNode getSubNode(char k) { // 得到指定位置的值  if (subNodes == null)  return null;  return subNodes.get(k);  }  public boolean isAlsoLeaf() { //判断是否为叶子节点  return false;  }  } |

|  |
| --- |
| 代码4-2：叶子节点类 |
| **class** LeafNode extends TreeNode {  int freq; /\*一个词的词频\*/  String word; /\*一个完整的词\*/  public LeafNode(TreeNode treeNode) {  super.subNodes = treeNode.getnodeMap();  }  public boolean isAlsoLeaf() {//判断是否为叶子节点  return true;  }  } |

构建完两个节点类之后，接下来就是利用这两个类完成字典树的构建过程。构建字典树就是把存储词库的文件读取并加入到内存的过程。具体的构建流程图如图4-9所示。



图4-9 字典树构建流程图

3 常用词语审校

常用词语审校的核心思想就是找出文本中的错误词语，然后再用词库中正确的词对该文本中的词语进行校对。这样如何发现文本中的错误词语就成了该部分的关键问题之一。本文采用的是n元组块后的词与词库中的词计算相似度的方法来查找文本中出现的错误的。词语审校过程的流程图如图4-10所示。以下分别介绍审校过程各个部分的工作：

（1）分词

分词就是利用分词系统对输入的文本进行分词。因为之后的所有处理过程都基于分词的结果，所以第一步首先就是要对输入的文本进行分词。

（2）组块

许多分词系统是按照词库来分词的，所以对于那些错词，很多情况下分词系统是无法将其分正确的。例如“项目名称”，正确的分词结果是“项目/n 名称/n”，若将其写为“项木名称”那分词系统则会将错误的“项木”分开，其结果为“项/q 木/ng 名称/n”，这是因为分词词库里面根本没有“项木”这个词。同样在审校的词库里面也没有“项木”这个词。若想要查找出错词“项木”就必须要进行组块。本文采用的组块策略是最长词优先的原则，先进行长词组块，再逐步缩短。具体步骤如下：

a）用该词与其后的两个词进行组块，然后再判断组合的块是否为常用词并检测是否有错，若有错则保存错词及其校对结果并跳过后面的步骤，并从其后的第三个词重新开始步骤a；

b）若步骤a没找到错误，则用该词与其后的一个词进行组块，然后再判断组合的块是否为常用词并检测是否有错，若有错则保存错词及其校对结果并跳过后面的步骤，并从其后的第二个词重新开始步骤a；

c）若步骤b还是没有找到错误，则直接检测当前词是否有错，若有错则保存错词及审校结果，然后从下一个词开始重新开始步骤a，直至所有词都已处理完。



图4-10 常用词语审校流程图

（3）字典树模糊查询

当待查词为正确的词的时候自然不需要模糊查询，可以直接按字逐层查询字典树即可得到查询结果。但若词语中存在一个或多个字出现错误时，这时候直接查询字典树无法找到与之对应的词。所以本文采用了字典树模糊查询的方法。字典树模糊查询的具体流程如图4-11所示。



图4-11 字典树模糊查询流程图

由图4-11可知，字典树模糊查询主要分为以下几步：

a）在字典树中查询待查词的第一个字，并建立了数据结构Map<trie, List<TreeNode>> map来保存查询结果。查询主要采用两种策略，首先是直接查询，若能在根节点的子节点中直接查询到该字，则将该字对应的节点保存到map中，若不能找到则遍历根节点的所有子节点，利用拼音和笔画相似度（分别如公式4-1和4-2所示）来查找与该字相似的字，并将所有相似的字所对应的节点保存到map中；

b）遍历map中相应层的List<TreeNode>，从每一个TreeNode的下一个节点来查询待查词的下一个字，查询策略还是与a中的查询策略一致，首先是直接查询，若能在该节点的子节点中直接查询到该字，则将该字对应的节点保存到map中，若不能找到则遍历该节点的所有子节点，利用拼音和笔画相似度（分别如公式4-1和4-2所示）来查找与该字相似的字，并将所有相似的字所对应的节点保存到相应层的map中，并从下一个字开始重复步骤b，直至所有字均被查找完成；

c）当待查词的所有字均已查找完毕，则遍历map中的最大key值（即最后一层）的List<TreeNode>，判断每一个节点是否为叶子节点，若为叶子节点则将叶子节点中的词保存到词列表（wordList）中去，遍历完成之后则返回wordList。

（4）审校结果获取

经过了步骤3，若其中的某个词在词库里面存在相同或相似的词的时候都会得到一个由一个或多个词组成的wordList。当wordList中只有一个词的时候比较简单，只需要比较两个词是否一致就行了，若一致忽略该处的词，若不一致，则将wordList中的词作为审校结果返回；但当wordList中有多个词的时候则需要将待审校词与wordList中的词逐一进行相似度计算。最后将相似度最大的词作为审校结果返回。

### 4.1.3 敏感词检测

敏感词一般指的是带有反执政党、暴力、不健康色彩的词或者不文明的用语。在很多网站都很注重敏感词的检测与屏蔽。本系统也设计了针对敏感词检测的方法。

本文的敏感词检测主要分为以下几个步骤：

1 词库构建

本文的敏感词词库是在网上直接下载的，里面包含492个敏感词。其中部分敏感词如表4-3所示。

表4-3 敏感词表示例

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 敏感词 | 编号 | 敏感词 |
| 1 | “六四”政治风波 | 5 | 台湾主权独立 |
| 2 | tmd | 6 | 藏独 |
| 3 | 台独 | 7 | 大跃进 |
| 4 | 台湾独立 | 8 | 呆子 |

2 字典树构建

本节的字典树还是采用的4.1.2中的字典树的结构，构建的流程也与上一节中的构建流程是一样的，唯一的区别在于加载的词库不同。

3 敏感词检测

敏感词主要就是检测待审校文本中是否存在敏感词，若存在给出相应的标识。本文采用的是顺序遍历待审校文本，然后对文本中的每一个字查找字典树的方式来检测文本中是否存在敏感词。设待审校文本的长度为*n*，因为字典树中采用的是HashMap，其key值为每个词的每个字，所以在字典树中查找每个字的时间复杂度为*O*(1)。这样就可以得出通过字典树来检测一个长度为n的文本中是否存在敏感词的时间复杂度就是遍历该文本的时间复杂度，即*O*(n)。敏感词具体检测的算法流程图如图4-12所示。



图4-12 敏感词检测流程图

## 4.2 标点符号审校

### 4.2.1 标点符号错误类型介绍

目前市场上的一些校对软件以及学术界很少提及对标点符号的校对问题研究，基本没有有效的方法提出。但这并不意味着标点符号校对问题无从下手，本文认为可以结合标点符号的习惯用法，提取出特定的规则来对标点符号进行审校。标点符号审校主要包括标点符号的误用、冗余等，现分类阐述如下。

1 标点符号误用

标点符号误用审校主要是对误用的标点符号进行审校，如分号误用为句号、句号误用为逗号及书名号的误用等。其中标点符号误用举例如下：

（1）非疑问句误用问号：

我不知道这条路谁能走通？但我一定要坚定不移地走下去。

（2）选择问句中间的停顿误用问号：

今天去呢？还是明天去呢？我实在是拿不定主意。

（3）倒装句中问号与叹号前置错误：

到底怎么办？这件事。

（4）分句之间误用顿号：

他留给我的印象依然如故：天真中透示着纯真、活泼中蕴含着稚气。

（5）联合词组不同层次的词语之间误用顿号：

上海的越剧、沪剧、淮剧、湖北的黄梅戏、河南的豫剧，在这次会演中，都带来了新剧目。

（6）句中未用逗号径直用分号：

任弼时同志一生三怕：一怕工作少；二怕麻烦人；三怕多用钱。

（7）句中已用句号再用分号：

一、学习贵在自觉。要有笨鸟先飞的精神，自我加压；二、学习贵在刻苦。要有锲而不舍的精神，持之以恒；……

2 标点符号冗余

标点符号重复审校主要是对重复的标点符号进行审校，如冒号的重复及“等等”与省略号的重复等。标点符号冗余主要包括以下几种情况：

（1）概数用顿号：

一天才走五、六里。

（2）集合词语用顿号：

如今，科学、技术的发展，必然会推动工、农业的发展。

（3）连词前用顿号：

观众长时间地等待，只为一睹她的风采、或签上一个名。

（4）短暂停顿用冒号：

他为什么总是说：“我疼你，我要保护你”之类的话？

（5）省略号和“等”“等等”同时使用：

一起去的有十七八个人，有我姐姐、表弟、表妹、同学……等。

（6）冒号中再用冒号：

昨天开校会，校长宣布：学校要实行两项改革措施：一是岗位工资制，二是岗位聘任制。

### 4.2.2 标点符号审校

1 正则表达是构建

要使用正则表达式校对文本中出现的标点符号，首先需要构建一个审校标点符号的正则表达式库。本文针对不同类型标点符号错误类型构建了34条正则表达式，其中部分正则表达式及其校对的类型如表4-4所示。其中每条规则的“@@”后面是对应的审校意见。

表4-4 标点符号校对正则表达式示例

|  |  |  |
| --- | --- | --- |
| 编号 | 模式串 | 类型 |
| 1 | [\u4e00-\u9fa5]? (…+)等@@ | 省略号与等不能连用 |
| 2 | [\u4e00-\u9fa5](：)[\u4e00-\u9fa5]+：. @@， | 冒号不能连续使用 |
| 3 | .”[\u4e00-\u9fa5]+[说道讲答问](，)[^“]. @@。 | 引用结束时要用句号 |
| 4 | [\u4e00-\u9fa5]?呢|吗(。) @@？ | 语气词后面一般不用句号 |
| 5 | [\u4e00-\u9fa5]+ (？{4,}) @@？ | 问号连用次数一般不大于三次 |
| 6 | [\u4e00-\u9fa5]+ (、)以?及@@， | 顿号不与“以”、“及”连用 |



图4-13 标点符号审校流程图

2 审校步骤

标点符号校对以段落为单位对标点符号进行扫描，将段落中出现的标点符号存入标点符号队列中，接着再遍历标点符号列表，根据标点符号类型选择对应的规则库对该段落进行校对，如果匹配到了则说明该段落中出现了此标点符号的误用错误，并根据规则返回所对应的解决方案与出错位置。对于括号与书名号等配对标点符号则采用堆栈进行校对，简言之就是如果扫描到了配对标点符号前向符号““”、“《”、“（”、“{”、“[”等则将其压栈，并记录此标点符号在段落中的位置。若为后向标点符号“””、“》”、“）”、“}”、“]”等则检验栈中是否有相应的前向标点，若有则将其从堆栈中弹出；如果不存在对应的前向标点，则将当前标点记为错误。具体标点符号的审校流程如图4-13所示。

## 4.3 数字审校

### 4.3.1 数字审校概述

数字审校主要是对日期、年龄等错误或不规范数字的审校，现分类阐述如下：

1 日期审校

日期审校主要是对错误日期或错误时间的审校，如识别并标注错误形式的年份（如“19999年”）或错误形式的时间（如“2007年2月29日”）等。

2 年龄审校

年龄审校主要是对不现实的年龄数据的审校，如210岁等。

3 号码审校

号码审校主要包括对有固定格式的号码的审校，如身份证号码、手机号码、电话号码、邮政编码、电子邮件、车牌号、IP地址等。

4 专用数字审校

专用数字审校主要是对具有固定书写格式的专用数字的审校，如将“3000里江山”校正为“三千里江山”等。

5 其它审校

其它审校主要包括对大小写数字的混用、全角半角混用及不同格式数字的审校等，如将“康熙18年”校正为“康熙十八年”，将“19７８年”校正为“1978年”等。

### 4.3.2 数字审校流程

数字审校基本采用的是通过正则表达式来检测并校对待审校文本中出现的数字。具体审校过程分为以下两个部分。

1 正则表达式构建

要使用正则表达式校对文本中出现的数字，首先需要构建一个正则表达式库。本文针对不同类型的数字格式构建了26条正则表达式，其中部分正则表达式及其校对的类型如表4-5所示。

表4-5 数字审校正则表达式示例

|  |  |  |
| --- | --- | --- |
| 编号 | 模式串 | 类型 |
| 1 | (?<hzdateymd>\d+(年|\-|\.|/)\d+(月|\-|\.|/)\d+(日|)) | 检验年月日是否有效 |
| 2 | (?<pemail>\w+([-+.]\w+)\*@\w+([-.]\w+)\*\.\w+([-.]\w+)\*) | 检验邮箱是否有效 |
| 3 | (?<ptel>((13[0-9])|(15[^4,\D])|(18[0,5-9]))\d{8}$) | 检验手机号是否有效 |
| 4 | (?<pip>(1\d{2}|2[0-4]\d|25[0-5]|[1-9]\d|[1-9])\.(1\d{2}|2[0-4]\d|25[0-5]|[1-9]\d|\d)\.(1\d{2}|2[0-4]\d|25[0-5]|[1-9]\d|\d)\.(1\d{2}|2[0-4]\d|25[0-5]|[1-9]\d|\d)$) | 检验ip地址是否有效 |
| 5 | (?<agehuman>\d+)(\D\*)(岁|出头|高龄) | 检验年龄 |
| 6 | (?<pbuscard>(冀|豫|云|辽|黑|湘|皖|鲁|新|苏|浙|赣|鄂|桂|甘|晋|蒙|陕|吉|闽|贵|粤|川|青|藏|宁|渝|京|津|沪|琼){1}[A-Z]{1}( |•|-){0,1}[A-Z\_0-9]{5}) | 检验车牌号是否有效 |



图4-14 数字审校流程图

2 数字审校

数字审校是采用正则表达式的规则来进行校对，其具体过程就是遍历正则表达式库，查找待审校文本中是否存在该正则表达式格式所对应的数字，若存在则判断其是否正确，若不正确则给出错误提示。例如“2007年2月29日”，首先利用正则表达式“(?<hzdateymd>\d+(年|\-|\.|/)\d+(月|\-|\.|/)\d+(日|))”将该日期找出来，因为2007年2月只有28天，并没有29日，所以提示该日期为错误日期。由于日期审校、年龄审校、号码审校等没有校对的依据，故只做正确性校对，没有修改建议。具体数字审校的流程如图4-14所示。

## 4.4 拼音审校

### 4.4.1 拼音审校概述

拼音审校包括不符合拼音规范的拼音的审校及大小写错误的审校，现分类阐述如下：

1 字符审校

字符审校是对汉语拼音字符错误的审校，除多音字导致的拼音错误审校外，还包括人名字符审校及地名字符审校等，主要是对专用的人名及专用或通用的地名拼音的审校，还包括其它原因导致的字符错误的审校，如将 “Běljīng（北京）”校正为“Běijīng”等

2 大小写审校

大小写审校是对汉语拼音大小写的审校，除句首大写及专用词语开头大写外，还包括人名大小写审校及地名大小写审校等，主要是对专用的人名及专用或通用的地名拼音中大小写的审校，如将“Lǐhuá（李华）”校正为“LǐHuá（李华）”、将“Běijīngshì（北京市）”校正为“Běijīng Shì（北京市）”等。

### 4.4.2 拼音审校流程

本文拼音审校主要针对的是文本中出现的拼音标注，如“北京（běijīng）”。审校过程主要是获取标注的拼音，然后用pinyin4j工具包来获取被标注拼音的正确拼音，然后在对其比较其拼音是否正确，若不正确给出用pinyin4j工具获取的拼音作为审校结果，拼音审校具体流程图如图4-15所示。



图4-15 拼音审校流程图

图4-15中的检测标注拼音的方法如下：

拼音审校的第一步就是检测待审校文本中是否存在标注拼音，检测方法是通过正则表达式“[(a-z)|(A-Z)]?(ā|á|ă|à|ē|é|ĕ|è|ī|í|ĭ|ì|ō|ó|ŏ|ò|ǖ|ǘ|ǚ|ǜ|ü|ū|ù|ú|ŭ|ù|)?[(a-z)|(A-Z)]?”来检测，若存在则利用规则"[^aoeiuü]?h?[iuü]?(ai|ei|ao|ou|er|ang?|eng?|ong|a|o|e|i|u|ng|n)?"来对检测出来的拼音进行切分，如“BĕijīngShì”,则可以切分为“Bĕi jīng Shì”，然后在根据切分出来的拼音个数在标注拼音标注前后来获取被标注拼音的汉字，用pinyin4j工具包分别获取两组词的拼音，并与转化为小写并切分好的标注拼音进行相似度计算，取相似度最高的词作为被标注拼音的汉字。如“北京市（BĕijīngShì）今天天气很好”，因为拼音被切分为三段，即为三个汉字的拼音，所以取该拼音前后的三个汉字，分别为“北京市”和“今天天”，接着获取这两个词的拼音分别为“bĕi jīng shì”和“jīn tiān tiān”，经计算二者与转换为小写并切分好的标注拼音“bĕi jīng shì”的相似度分别为1.0和0.32。所以最终取“北京市”作为被标注拼音的汉字。这里所用的相似度算法还是公式(4-1)中所提的最长公共子序列计算相似度的方法。

## 4.5 本章小结

本章重点介绍了审校系统中的词语审校功能的实现算法和流程，从实体名称审校、常用词语审校以及敏感词检测三个方面介绍了词语审校，其中实体名称审校采用的是实体链接的方法来实现的，而常用词语审校和敏感词检测则是采用构建字典树来实现的，二者区别在于常用词语审校则是采用模糊匹配的方法来查找词库，而敏感词则是采用精确查找的方式来查找词库，而且二者所加载的词库也不相同。本章还介绍了标点符号、数字审校和拼音审校的方法，对于这三类错误，本文采取的是构建正则表达式的方法来实现的。

# 第5章 系统实现与演示

## 5.1 前言

本章主要介绍中文文本自动系统web框架的实现技术以及每个前台功能的操作流程及展示。

## 5.2 系统框架

本文采用的Spring MVC框架来实现中文文本自动校对系统的Web框架。Spring MVC是基于请求驱动的Web框架，用户提交的请求通过DispatcherServlet来通过查询处理器映射器来确定由哪个控制器来处理。所以搭建Spring MVC框架首先要在web.xml中配置DispatcherServlet配置文件，具体配置如下：

|  |
| --- |
| <servlet>  <servlet-name>proofread</servlet-name>  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>  <init-param>  <param-name>contextConfigLocation</param-name>  <param-value>/WEB-INF/proofread-servlet.xml</param-value>  </init-param>  <load-on-startup>1</load-on-startup>  </servlet> |

在web.xml中配置好了DispatcherServlet配置文件后，接着就是在DispatcherServlet配置文件中配置DispatcherServlet。在DispatcherServlet配置文件中需要配置以下几部分内容：

（1）组件扫描基础包，其中包含各个功能的控制器；

（2）资源解析器，负责解析系统所需资源所在位置；

（3）自动扫描组件，开启自动扫描组件后，DispatcherServlet会根据注解自动适配控制器；

（4）视图解析器，负责解析model返回的视图逻辑名称，并将其转换为特定的视图名称；

（5）文件解析器，负责解析文件上传及下载。

其具体配置如下所示：

|  |
| --- |
| <!-- 组件扫描包 -->  <context:component-scan base-package="com.swjtu.controller" />  <!—自动扫描组件 -->  <mvc:annotation-driven />  <!—资源解析器 -->  <mvc:resources mapping="/members/\*\*" location="/WEB-INF/members/" />  <mvc:resources mapping="/assets/\*\*" location="/WEB-INF/assets/" />  <!—视图解析器 -->  <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">  <property name="prefix">  <value>/WEB-INF/views/</value>  </property>  <property name="suffix">  <value>.jsp</value>  </property>  </bean>  <!—文件解析器-->  <bean id="multipartResolver" class="org.springframework.web.multipart.commons.CommonsMultipartResolver" p:defaultEncoding="UTF-8" p:maxUploadSize="5400000" p:uploadTempDir="files">  </bean> |

## 5.3 系统功能展示

### 5.2.1 注册

用户注册是该系统的一个基本功能，用户首先在图5-1所示的注册页面填写注册所需的基本信息，这些都是必不可少的信息，邮箱需要用于邮箱验证，而密码则是用户登录系统的凭证；填写完邮箱和密码之后点击注册，系统会给注册邮箱发一份验证邮件如图5-2所示，用户点击验证邮件中的链接即可完成注册进入系统，即用户个人信息维护界面，如图5-3所示。



图5-1 用户注册



图5-2 邮箱验证邮件



图5-3 个人信息维护

### 5.2.2 登录

用户已经完成注册后可以随时登录中文文本自动校对系统，用户只需要在如5-4所示的登录界面输入正确的邮箱和密码即可登录到该系统。



图5-4 用户登录

### 5.2.3 个人信息维护

用户刚刚完成注册后进入该系统，用户的基本信息除了邮箱之外其他的都是空的如图5-3所示，用户可以在各项基本信息框内输入自己的基本信息，然后点保存即可，修改后的基本信息如图5-5所示。用户还可以在该界面修改密码，只要验证了原密码即可对密码进行修改。



图5-5 修改个人信息及密码

### 5.2.4 在线审校

在线审校是该系统的主要功能之一，用户进入在线审校界面后可以在左边的输入框内输入要审校的文本，然后点击审校，待后台完成审校后在右侧校对结果框内会输出后台校对的审校的结果，其中红色的代表错误或者出现疑似错误的地方。如图5-6所示，其中第一条“形像代言人”中的“形像”二字为错词，正确的应该是“形象”，系统会给出校对结果以及原文本供用户选择。



图5-6 在线审校

### 5.2.5 离线审校

离线审校主要针对的是word文档的审校，用户进入离线审校界面后可以选择要上传的word文件，然后点击上传，如图5-7所示。用户完成文件上传之后，word文档交由系统后台去审校，待用户审校完成之后会在离线审校界面显示出来审校完成后的文件供用户下载，用户只需要点击文件名后的下载链接即可完成审校完成后的结果的下载，如图5-9所示。

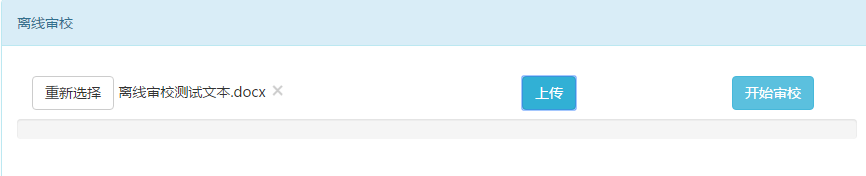


图5-7 文件上传

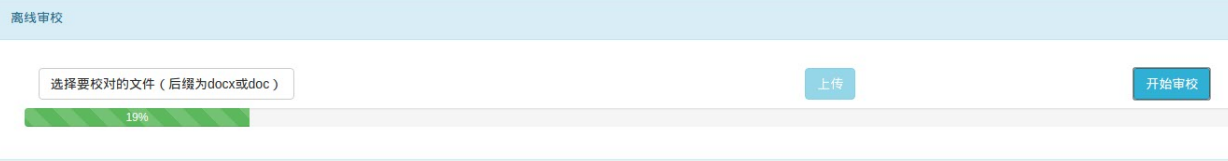
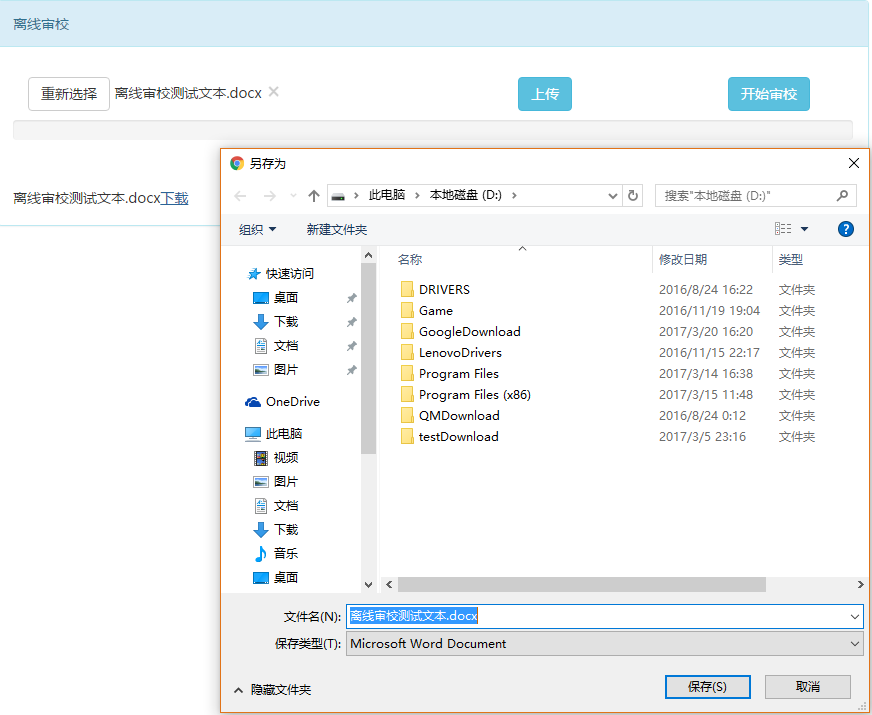


图5-8 离线审校



如5-9 离线审校结果下载

## 5.3 本章小节

本章主要介绍了如何使用Spring MVC搭建中文文本自动校对系统的Web框架，并详细的介绍了注册、登录、个人信息维护、在线审校以及离线审校的操作流程。重点介绍了每个功能的操作流程。

# 结 论

## 本文工作总结

互联网信息快速增长的同时也造成了网络文本的参差不齐，而目前大多数审校工作特别是在新闻出版和电视广播等行业仍然是以人工审校为主。本文针对这一现象构建了一个基于B/S模式的中文文本自动校对系统。主要研究内容包括如下几个方面：

1. 针对中文文本自动校对系统展开系统需求分析，详细分析系统的用户需求，业务需求以及功能需求。

2. 用MVC框架对系统框架进行设计，详细地分析系统中每一层所包含的功能，并详细地分析并设计了审校服务的具体功能。

3. 实现中文文本自动校对系统的后台功能，其中包括采用CRF加分词的技术识别文本中的实体并采用实体链接的技术来实现词语审校中的实体名称审校，而词语审校中的常用词语审校即敏感词审校则采用构建字典树的方式来实现；针对标点符号和数字审校主要采用的是构建规则库来实现；而对于拼音审校本文利用pinyin4j工具包来获取正确词的拼音，再与原文中的标注的拼音比较来进行审校的。

4. 利用Spring MVC框架来实现中文文本自动校对系统的web框架。其中包括前台Jsp界面的编写，控制层逻辑的实现。

## 未来工作展望

本文提出的审校方法虽然可以审校出文本中很多常见错误，也能给出相应的修改意见，但是局限于时间和作者本身的能力，该系统的审校方法还有很大改进空间。接下来的工作可以从以下几个方面来进行：

（1）本文的词语审校针对单个词的错误利用词典的方式来校对的，并没有考虑句子的语义。下阶段可以从语义的角度来考虑词语审校，根据句子语义来选择合适的词对错误文本进行审校。方法则可以尝试一下用深度学习模型。

（2）本文的标点符号审校、数字审校都是采用规则来进行的，下阶段可以尝试寻找合适的机器学习或者深度学习模型来解决这两部分的审校工作。

# 致 谢

光阴荏苒，日月如梭，转眼间三年的硕士生涯就即将结束，但在内心还是有诸多的不舍与留恋。

首先，我要由衷的感谢我的导师李天瑞教授，感谢李老师一直以来对我生活和学习上的无私的指导和帮助。李老师每周会不定期询问我的生活和学习状况。每当我遇到困惑和难题时，李老师总能敏锐的发现问题的关键点，并会很耐心的给予我详细的指导。李老师对待工作和学术严谨认真的态度以及对新知识的明锐洞察力对我影响极大，使我受益终身。同时我还要感谢我的指导老师贾真，我从研一开始就进入了贾老师带的自然语言处理组，这三年贾老师对我学习和生活都给予了极大的帮助。

其次我还要感谢上一届的师兄黄峻福，他和我都在自然语言处理组，他在校期间给予了学习上很大的帮助，每当我有问题找他时他都会无私的帮助我。感谢博士后刘胜久师兄，他在我硕士三年期间对我无私的帮助和细心的指导。感谢我们自然语言处理组的其他同学，和你们一起做比赛和项目使我获益良多。

感谢我的父母，是你们一直以来对我默默的支持和付出才能让我专心于学习。你们的爱和鼓励使我不断前行，无所畏惧。

感谢各位评审和答辩的各位老师，谢谢你们能够在百忙之中抽出时间来对我的论文进行审阅和答辩，感谢你们提出的宝贵的意见。

# 参考文献

1. Adamov A. Distributed file system as a basis of data-intensive computing[C]. 2012 6th International Conference on Application of Information and Communication Technologies (AICT), IEEE, 2012: 1-3.
2. Peterson J L. Computer programs for detecting and correcting spelling errors[J]. Communications of the ACM, 1980, 23(12): 676-687.
3. Kukich, Karen. Techniques for automatically correcting words in text. ACM Computing Surveys (CSUR), 1992, 24(4): 377-439.
4. Pollock J J, Zamora A. Automatic spelling correction in scientific and scholarly text[J]. Communications of the ACM, 1984, 27(4): 358-368.
5. Riseman E M, Hanson A R. A contextual postprocessing system for error correction using binary n-grams[J]. Computers, IEEE Transactions on, 1974, 100(5): 480-493.
6. Riseman E M, Ehrich R W. Contextual word recognition using binary digrams[J]. Computers, IEEE Transactions on, 1971, 100(4): 397-403.
7. Golding A R, Schabes Y. Combining trigram-based and feature-based methods for context-sensitive spelling correction[C]. Proceedings of the 34th annual meeting on Association for Computational Linguistics. Association for Computational Linguistics, 1996: 71-78.
8. Evershed, John, and Kent Fitch. Correcting noisy OCR: Context beats confusion. Proceedings of the First International Conference on Digital Access to Textual Cultural Heritage. ACM, 2014, 45-51.
9. Kumar, Atul, and Gurpreet Singh Lehal. Automatic Text Correction for Devanagari OCR. Indian Journal of Science and Technology, 2016, 9(45).
10. Bulatov, Konstantin, et al. Trigram-based algorithms for OCR result correction. Proceedings in Ninth International Conference on Machine Vision. International Society for Optics and Photonics, 2017, 103410O-103410O.
11. 于勐, 姚天顺. 一种混合的中文文本校对方法[J]. 中文信息学报, 1998, 12(2): 31-36.
12. 李建华, 王晓龙. 多特征的中文文本校对算法的研究[J]. 计算机工程与科学, 2001, 23(3): 93-96.
13. 陈笑蓉, 秦进, 汪维家, 等. 中文文本校对技术的研究与实现[J]. 计算机科学, 2003, 30(11): 53-55.
14. 龚小谨, 罗振声. 中文文本自动校对中的语法错误检查[J]. 计算机工程与应用, 2003, 39(8): 98-100.
15. 骆卫华, 罗振声, 龚小谨. 中文文本自动校对的语义级查错研究[J]. 计算机工程与应用, 2003, 39(12): 115-118.
16. 骆卫华, 罗振声, 龚小瑾. 中文文本自动校对技术的研究[J]. 计算机研究与发展, 2004, 41(1): 244-249.
17. 王斯宇, 邵波. 基于CSSCI的文本自动校对系统的构建与实现[J]. 高校图书馆工作, 2014, 34(6): 50-54.
18. 张仰森, 唐安杰, 张泽伟. 面向政治新闻领域的中文文本校对方法研究[J]. 中文信息学报, 2014, 28(6): 79-84.
19. Lafferty, John, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. Proceedings of the eighteenth international conference on machine learning, ICML. 2001. 1(1). 282-289.
20. Yao, Kaisheng, et al. Recurrent conditional random field for language understanding. Proceedings of Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on. 2014, 4077-4081.
21. Ammar, Waleed, Chris Dyer, and Noah A. Smith. "Conditional random field autoencoders for unsupervised structured prediction." Advances in Neural Information Processing Systems. 2014:3311-3319.
22. 张盛, 李芳. 基于迭代两步 CRF模型的评价对象与极性抽取研究[J]. 中文信息学报, 2015, 29(1): 163-169.
23. 栗伟, 赵大哲, 李博, 等. CRF与规则相结合的医学病历实体识别[J]. 计算机应用研究, 2015, 4, 1082-1086.
24. Uzunbas, Mustafa Gokhan, Chao Chen, and Dimitris Metaxas. "An efficient conditional random field approach for automatic and interactive neuron segmentation." Medical image analysis 2016, 27: 31-44.
25. Vemulapalli, Raviteja, et al. Gaussian conditional random field network for semantic segmentation. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, 3224-3233.
26. 吴胜远. 一种汉语分词方法. 计算机研究与发展, 1996, 33(4): 306-311.
27. Howell, Peter, Stevie Sackin, and Kazan Glenn. Development of a Two-Stage Procedure for the Automatic Recognition of Dysfluencies in the Speech of Children Who StutterII. ANN Recognition of Repetitions and Prolongations With Supplied Word Segment Markers. Journal of Speech, Language, and Hearing Research, 1997, 40(5): 1085-1096.
28. 黄昌宁. 中文信息处理中的分词问题. 语言文字应用, 1997, 1: 74-80.
29. Xianghao, Guo, Zhong Yixin, and Yang Li. A Fast Algorithm for Chinese Words Automatic Segment Based on Two letters word family Structure. Journal of the China Society for Scientific and Technical Information, 1998, 05.
30. 孙茂松, 左正平, 黄昌宁. 汉语自动分词词典机制的实验研究. 中文信息学报 2000, 14(1): 1-6.
31. Dong, W. U. Word Segment and Search Techniques for Chinese Information Search Engines [J]. Computer Applications 7, 2004.
32. Bernhard, Delphine. Unsupervised morphological segmentation based on segment predictability and word segments alignment. Proceedings of 2nd Pascal Challenges Workshop. 2006, 19-24.
33. Siwei, Lai, et al. Chinese Word Segment Based on Character Representation Learning. Journal of Chinese information processing, 2013, 27(5): 8-14.
34. 来斯惟, 徐立恒, 陈玉博, 等. 基于表示学习的中文分词算法探索. 中文信息学报, 2013, 27(5): 8-15.
35. 尹存燕, 黄书剑, 戴新宇, 等. 中英命名实体识别及对齐中的中文分词优化. 电子学报, 2015, 43(8), 1481-1487.
36. 刘开瑛.中文文本自动分词和标注. 商务印书馆, 2000.
37. 黄昌宁, 高剑峰, 李沐. 对自动分词的反思. 语言计算与基于内容的文本处理 2003.
38. Chen, Xinchi, et al. Long Short-Term Memory Neural Networks for Chinese Word Segmentation. EMNLP. 2015:197-1206.
39. Derczynski L, Maynard D, Rizzo G, et al. Analysis of named entity recognition and linking for tweets[J]. Information Processing & Management, 2015, 51(2): 32-49.
40. Prokofyev R, Demartini G, Cudré-Mauroux P. Effective named entity recognition for idiosyncratic web collections[C]. Proceedings of the 23rd international conference on World wide web. ACM, 2014: 397-408.
41. Konkol M, Brychcín T, Konopík M. Latent semantics in Named Entity Recognition[J]. Expert Systems with Applications, 2015, 42(7): 3470-3479.
42. Yao T, Ding W, Erbach G. Correcting word segmentation and part-of-speech tagging errors for Chinese named entity recognition[M]. The Internet Challenge: Technology and Applications. Springer Netherlands, 2002: 29-36.
43. Han X, Sun L, Zhao J. Collective entity linking in web text: a graph-based method[C]. Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval. ACM, 2011: 765-774.
44. Hachey B, Radford W, Nothman J, et al. Evaluating entity linking with Wikipedia[J]. Artificial intelligence, 2013, 194: 130-150.
45. Demartini G, Difallah D E, Cudré-Mauroux P. ZenCrowd: leveraging probabilistic reasoning and crowdsourcing techniques for large-scale entity linking[C]. Proceedings of the 21st international conference on World Wide Web. ACM, 2012: 469-478.
46. Lin T, Etzioni O. Entity linking at web scale[C]. Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction. Association for Computational Linguistics, 2012: 84-88.
47. Bunescu R C, Pasca M. Using Encyclopedic Knowledge for Named entity Disambiguation[C]. In EACL. 2006, 6: 9-16.
48. Han X, Zhao J. Named entity disambiguation by leveraging wikipedia semantic knowledge[C]. Proceedings of the 18th ACM conference on Information and knowledge management. ACM, 2009: 215-224.
49. Zheng Z, Si X, Li F, et al. Entity disambiguation with freebase[C]. Proceedings of the The 2012 IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technology-Volume 01. IEEE Computer Society, 2012: 82-89.
50. Alhelbawy A, Gaizauskas R. Graph ranking for collective named entity disambiguation[C]. Annual Meeting of the Association for Computational Linguistics. 2014: 75-80.
51. Nguyen D B, Hoffart J, Theobald M, et al. Aida-light: High-throughput named-entity disambiguation[J]. Linked Data on the Web at WWW2014, 2014.

# 攻读硕士学位期间发表的论文及科研成果

**参与的科研项目**

[1] 国际横向项目：语义分析. 起止时间：2014.6-2015.6

[2] 国家自然科学基金项目：基于粒计算的多源异构动态数据挖掘关键技术研究（项目编号：61573292）. 起止时间：2016.1-2019.12.

# 第5章 系统实现与演示

## 5.1 前言

本章主要介绍中文文本自动系统web框架的实现技术以及每个前台功能的操作流程及展示。

## 5.2 系统框架

本文采用的Spring MVC框架来实现中文文本自动校对系统的Web框架。Spring MVC是基于请求驱动的Web框架，用户提交的请求通过DispatcherServlet来通过查询处理器映射器来确定由哪个控制器来处理。所以搭建Spring MVC框架首先要在web.xml中配置DispatcherServlet配置文件，具体配置如下：

|  |
| --- |
| <servlet>  <servlet-name>proofread</servlet-name>  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>  <init-param>  <param-name>contextConfigLocation</param-name>  <param-value>/WEB-INF/proofread-servlet.xml</param-value>  </init-param>  <load-on-startup>1</load-on-startup>  </servlet> |

在web.xml中配置好了DispatcherServlet配置文件后，接着就是在DispatcherServlet配置文件中配置DispatcherServlet。在DispatcherServlet配置文件中需要配置以下几部分内容：

（1）组件扫描基础包，其中包含各个功能的控制器；

（2）资源解析器，负责解析系统所需资源所在位置；

（3）自动扫描组件，开启自动扫描组件后，DispatcherServlet会根据注解自动适配控制器；

（4）视图解析器，负责解析model返回的视图逻辑名称，并将其转换为特定的视图名称；

（5）文件解析器，负责解析文件上传及下载。

其具体配置如下所示：

|  |
| --- |
| <!-- 组件扫描包 -->  <context:component-scan base-package="com.swjtu.controller" />  <!—自动扫描组件 -->  <mvc:annotation-driven />  <!—资源解析器 -->  <mvc:resources mapping="/members/\*\*" location="/WEB-INF/members/" />  <mvc:resources mapping="/assets/\*\*" location="/WEB-INF/assets/" />  <!—视图解析器 -->  <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">  <property name="prefix">  <value>/WEB-INF/views/</value>  </property>  <property name="suffix">  <value>.jsp</value>  </property>  </bean>  <!—文件解析器-->  <bean id="multipartResolver" class="org.springframework.web.multipart.commons.CommonsMultipartResolver" p:defaultEncoding="UTF-8" p:maxUploadSize="5400000" p:uploadTempDir="files">  </bean> |

## 5.3 系统功能展示

### 5.2.1 注册

用户注册是该系统的一个基本功能，用户首先在图5-1所示的注册页面填写注册所需的基本信息，这些都是必不可少的信息，邮箱需要用于邮箱验证，而密码则是用户登录系统的凭证；填写完邮箱和密码之后点击注册，系统会给注册邮箱发一份验证邮件如图5-2所示，用户点击验证邮件中的链接即可完成注册进入系统，即用户个人信息维护界面，如图5-3所示。



图5-1 用户注册



图5-2 邮箱验证邮件



图5-3 个人信息维护

### 5.2.2 登录

用户已经完成注册后可以随时登录中文文本自动校对系统，用户只需要在如5-4所示的登录界面输入正确的邮箱和密码即可登录到该系统。



图5-4 用户登录

### 5.2.3 个人信息维护

用户刚刚完成注册后进入该系统，用户的基本信息除了邮箱之外其他的都是空的如图5-3所示，用户可以在各项基本信息框内输入自己的基本信息，然后点保存即可，修改后的基本信息如图5-5所示。用户还可以在该界面修改密码，只要验证了原密码即可对密码进行修改。



图5-5 修改个人信息及密码

### 5.2.4 在线审校

在线审校是该系统的主要功能之一，用户进入在线审校界面后可以在左边的输入框内输入要审校的文本，然后点击审校，待后台完成审校后在右侧校对结果框内会输出后台校对的审校的结果，其中红色的代表错误或者出现疑似错误的地方。如图5-6所示，其中第一条“形像代言人”中的“形像”二字为错词，正确的应该是“形象”，系统会给出校对结果以及原文本供用户选择。



图5-6 在线审校

### 5.2.5 离线审校

离线审校主要针对的是word文档的审校，用户进入离线审校界面后可以选择要上传的word文件，然后点击上传，如图5-7所示。用户完成文件上传之后，word文档交由系统后台去审校，待用户审校完成之后会在离线审校界面显示出来审校完成后的文件供用户下载，用户只需要点击文件名后的下载链接即可完成审校完成后的结果的下载，如图5-9所示。

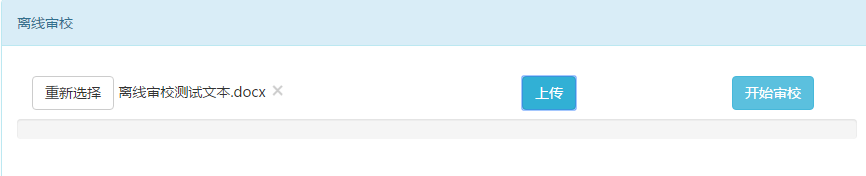


图5-7 文件上传

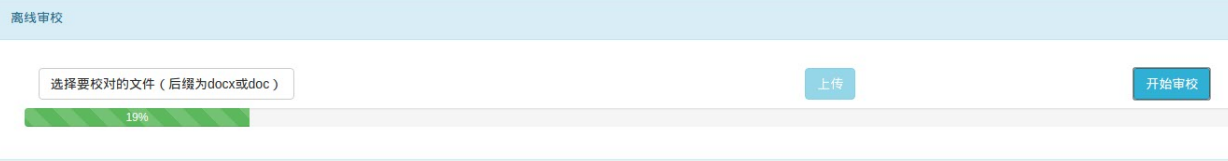
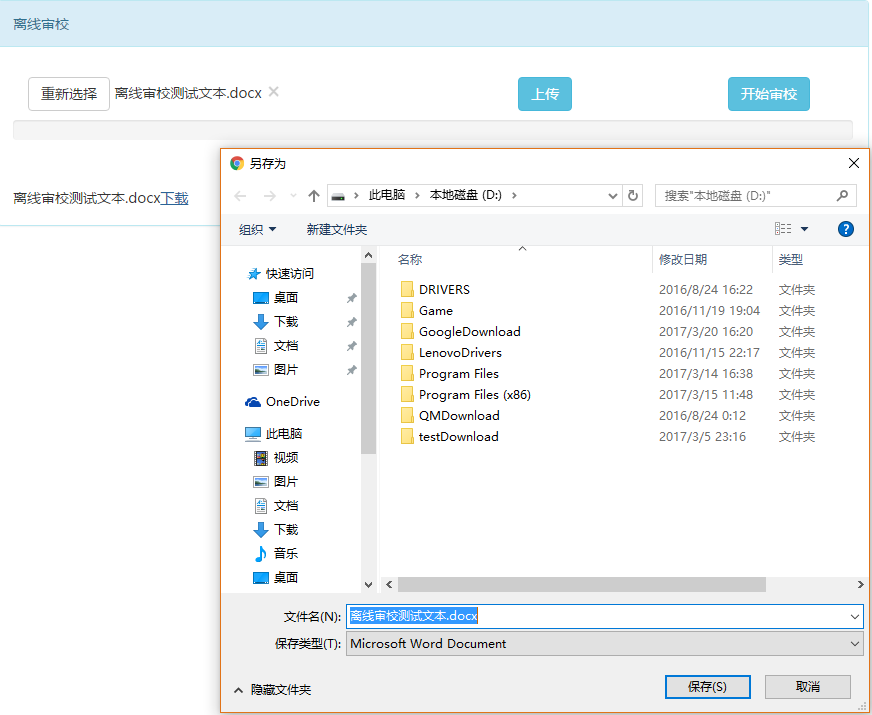


图5-8 离线审校



如5-9 离线审校结果下载

## 5.3 本章小节

本章主要介绍了如何使用Spring MVC搭建中文文本自动校对系统的Web框架，并详细的介绍了注册、登录、个人信息维护、在线审校以及离线审校的操作流程。重点介绍了每个功能的操作流程。

# 结 论

## 本文工作总结

互联网信息快速增长的同时也造成了网络文本的参差不齐，而目前大多数审校工作特别是在新闻出版和电视广播等行业仍然是以人工审校为主。本文针对这一现象构建了一个基于B/S模式的中文文本自动校对系统。主要研究内容包括如下几个方面：

1. 针对中文文本自动校对系统展开系统需求分析，详细分析系统的用户需求，业务需求以及功能需求。

2. 用MVC框架对系统框架进行设计，详细地分析系统中每一层所包含的功能，并详细地分析并设计了审校服务的具体功能。

3. 实现中文文本自动校对系统的后台功能，其中包括采用CRF加分词的技术识别文本中的实体并采用实体链接的技术来实现词语审校中的实体名称审校，而词语审校中的常用词语审校即敏感词审校则采用构建字典树的方式来实现；针对标点符号和数字审校主要采用的是构建规则库来实现；而对于拼音审校本文利用pinyin4j工具包来获取正确词的拼音，再与原文中的标注的拼音比较来进行审校的。

4. 利用Spring MVC框架来实现中文文本自动校对系统的web框架。其中包括前台Jsp界面的编写，控制层逻辑的实现。

## 未来工作展望

本文提出的审校方法虽然可以审校出文本中很多常见错误，也能给出相应的修改意见，但是局限于时间和作者本身的能力，该系统的审校方法还有很大改进空间。接下来的工作可以从以下几个方面来进行：

（1）本文的词语审校针对单个词的错误利用词典的方式来校对的，并没有考虑句子的语义。下阶段可以从语义的角度来考虑词语审校，根据句子语义来选择合适的词对错误文本进行审校。方法则可以尝试一下用深度学习模型。

（2）本文的标点符号审校、数字审校都是采用规则来进行的，下阶段可以尝试寻找合适的机器学习或者深度学习模型来解决这两部分的审校工作。

# 致 谢

光阴荏苒，日月如梭，转眼间三年的硕士生涯就即将结束，但在内心还是有诸多的不舍与留恋。

首先，我要由衷的感谢我的导师李天瑞教授，感谢李老师一直以来对我生活和学习上的无私的指导和帮助。李老师每周会不定期询问我的生活和学习状况。每当我遇到困惑和难题时，李老师总能敏锐的发现问题的关键点，并会很耐心的给予我详细的指导。李老师对待工作和学术严谨认真的态度以及对新知识的明锐洞察力对我影响极大，使我受益终身。同时我还要感谢我的指导老师贾真，我从研一开始就进入了贾老师带的自然语言处理组，这三年贾老师对我学习和生活都给予了极大的帮助。

其次我还要感谢上一届的师兄黄峻福，他和我都在自然语言处理组，他在校期间给予了学习上很大的帮助，每当我有问题找他时他都会无私的帮助我。感谢博士后刘胜久师兄，他在我硕士三年期间对我无私的帮助和细心的指导。感谢我们自然语言处理组的其他同学，和你们一起做比赛和项目使我获益良多。

感谢我的父母，是你们一直以来对我默默的支持和付出才能让我专心于学习。你们的爱和鼓励使我不断前行，无所畏惧。

感谢各位评审和答辩的各位老师，谢谢你们能够在百忙之中抽出时间来对我的论文进行审阅和答辩，感谢你们提出的宝贵的意见。