北京航空航天大學
BEIHANG UNIVERSITY

《基于知识图谱的自动问答系统》
设计说明书

北京航空航天大学

2016-01

版本变更历史

版本	提交日期	主要编制人	审核人	版本说明
1	2015.12.31	方凯	杨东东	完成 1、2(部分)、3(部分)、
2	2015.01.01	杨东东		完成 2(部分)、3(部分) 、4、5、6(部分)、7
3				6

目 录

1. 范围

1.1 标识

文档标识号：A2015-00-02-00

文档标题：<<基于知识图谱的自动问答系统>>--系统设计说明书

版本号：1.0

1.2 系统概述

本条应简述本文档适用的系统和软件的用途，它应描述系统和软件的一般特性；概述系统开发、运行和维护的历史；标识项目的投资方、需方、用户、开发方和支持机构；标识当前和计划的运行现场。

基于知识图谱的自动问答系统，是以中文为载体的系统，其数据库为以百度百科、维基百科、互动百科为主，运用其中的知识性信息进行人机交互以达到自动问答的目的的系统。采用目前发展中的实体分词技术、实体消歧技术、语法分析技术、语义分析技术等作为基础，综合开发而成。

系统尚在开发过程中；

投资方：无；

需方：用户；

用户：有提问需求者；

开发方：杨东东，李睿霖，方凯；

支持机构：提供知识库的单位；

运行现场：Android 系统。

1.3 文档概述

本文档用于阐述系统概况以及系统设计说明。该文档使用时保密性一般，由于未涉及核心代码，使用时可以半公开。

1.4 术语和缩略词

a.数据库(Database)：

MySQL: 一种关联数据库管理系统

Neo4j:一个高性能的,NOSQL 图形数据库

Redis:一个 key-value 存储系统，以超高效的查找存储著称，并具有数据持久的特性

b.自然语言处理(NLP)：

Entity Linking:实体链接

Page Rank:Google 开源的一个搜索算法

Entity Ambiguation:实体歧义

Trie Tree:前缀树

kNN: k 近邻算法 (kNN，k-NearestNeighbor)

LSA: 隐式语义分析 (Latent Semantic Analysis)

Markov Model:马尔科夫模型

Lucene: 一个开放源代码的全文检索引擎工具包，是一个全文检索引擎的架构

FudanNLP：一个中国国内做得还算不错的 NLP 处理开源包

c.矩阵论(Matrix)：

PCA:主成分分析，用于矩阵维度的降维方法 (Principal Component Analysis)

SVD:矩阵奇异值分解 (singular value decomposition method)

2. 引用文档

a.书籍包括：

《软件项目管理》 朱少民，韩莹 编著，人民邮电出版社

《软件项目管理》 Rajeev T Shandilya 编著 科学出版社

b.本项目的经核准的计划任务书和合同、上级机关的批文:

第九届《大学生创新创业训练计划》

c.引用资料：

《Open Question Answering Over Curated and Extracted Knowledge Bases》

《syntactic constraints on paraphrases extracted from parallel corpora》

《基于维基百科的自动词义消歧方法》 史天艺

《一个中文实体链接语料库的建设》 舒佳根

《基于角色标注的中国人名自动识别研究》张华平 刘群

《基于层叠隐马尔可夫模型的中文命名实体识别》俞鸿魁 张华平 刘群

《基于角色标注的中文机构名识别》俞鸿魁 张华平 刘群

《基于最大熵的依存句法分析》 辛霄 范士喜 王轩 王晓龙

《社区问答系统中若干关键问题研究》 廉鑫

3. 需求概述

概述系统的特性和需求，扩充软件需求说明中的信息，给出增加的细节，详尽地指出对系统需求规格说明中有关特性和需求作出的变更。

说明：给出新增功能的用例模型以及用例流程说明、类图及说明

功能需求：

1、能够实现定时的对网络的信息进行爬取，以更新数据库，同时将常用的数据放在系统的内存中进行运行查询删除。

2、能够实现对所得数据的训练，训练包括以下几个方面:

一个是搜索引擎高效索引文件的训练;

一个是分词模型的训练（主要为 CRF 条件随机场的模板数据的训练）;

一个是多层神经网络模型的训练（主要为聊天语料库）;

一个是最大完全子图模型的训练（主要为语法模板的抽取作铺垫）;

3、 能够与用户进行聊天；

4、 能够回答事实型问题、列举型问题、定义型问题和交互型问题等

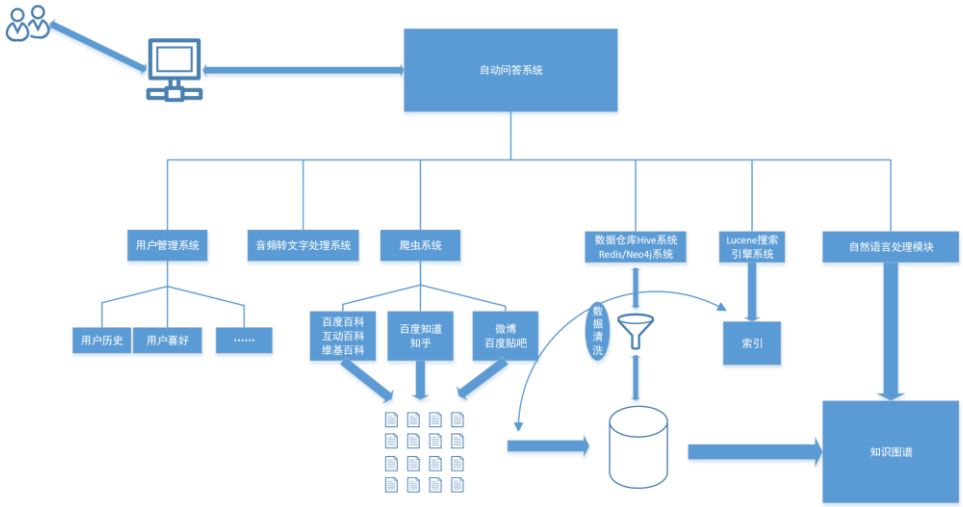
4. 体系结构设计

4.1 总体结构

用一系列图表给出系统的总体结构，并对图中的相关内容进行说明。可能包括软件体系结构、硬件体系结构、技术体系结构、支撑体系（部署和实施方案）结构等各个方面，可根据实际情况每个方面分为一小节来写。

对于软件体系结构描述，如果开发采用的是结构化分析与设计方法，建议给出模块组成及之间的调用关系、模块间的接口描述，说明:给出每个模块的具体功能；如果采用面向对象分析与设计方法，建议从 UML 的 4+1 视图描述软件结构：完善用例图、活动图，给出类图、时序图、状态图。

4.1.1 总体结构



4.1.2 用例图

用例图

4.1.3 活动图

活动图

4.1.4 类图

类图

4.1.5 时序图

状态图

4.1.6 状态图

状态图

4.2 功能分配

说明需求文档当中各项功能要求同总体结构的对应关系。

其中，

信息搜集功能对应于爬虫系统；

信息检索功能对英语基于 PageRank 的 Lucene 搜索引擎系统；

信息处理功能对应于分词系统、实体消歧系统；

信息存储功能对应于图数据库系统等

4.3 关键问题及解决方案

给出系统总体结构中所涉及的关键问题的设计决策和解决思路。

主要例举核心模板自然语言处理中最难的功能的关键问题，问题的具体解决思路见源码及参考文献。

4.3.1 新词发现

从文件集合中挖掘出内涵的新词语列表，可以用于用户专业词典的编撰；还可以进一步编辑标注，导入分词词典中，从而提高分词系统的准确度，并适应新的语言变化。

4.3.2 分词标注

对原始语料进行分词、自动识别人名地名机构名等未登录词、新词标注以及词性标注。并可在分析过程中，导入用户定义的词典。

4.3.3 统计分析与术语翻译

针对切分标注结果，系统可以自动地进行一元词频统计、二元词语转移概率统计（统计两个词左右连接的频次即概率）。针对常用的术语，会自动给出相应的英文解释。

4.3.4 大数据聚类及热点分析-Cluster

能够从大规模数据中自动分析出热点事件，并提供事件话题的关键特征描述。同时适用于长文本和短信、微博等短文本的热点分析。

4.3.5 大数据分类过滤

针对事先指定的规则和示例样本，系统自动从海量文档中筛选出符合需求的样本。

4.3.6 自动摘要-Summary

能够对单篇或多篇文章，自动提炼出内容的精华，方便用户快速浏览文本内容。

4.3.7 关键词提取-KeyExtract

能够对单篇文章或文章集合，提取出若干个代表文章中心思想的词汇或短语，可用于精化阅读、语义查询和快速匹配等。

4.3.8 文档去重-RedupRemover

能够快速准确地判断文件集合或数据库中是否存在相同或相似内容的记录，同时找出所有的重复记录。

4.3.9 HTML 正文提取-HTMLPaser

自动剔除导航性质的网页，剔除网页中的 HTML 标签和导航、广告等干扰性文字，返回有价值的正文内容。适用于大规模互联网信息的预处理和分析。

4.3.10 编码自动识别与转换

自动识别文档内容的编码，并进行自动转换，目前支持 Unicode/BIG5/UTF-8 等编码自动转换为简体的 GBK，同时将繁体 BIG5 和繁体 GBK 进行繁简转化。

5. 接口设计

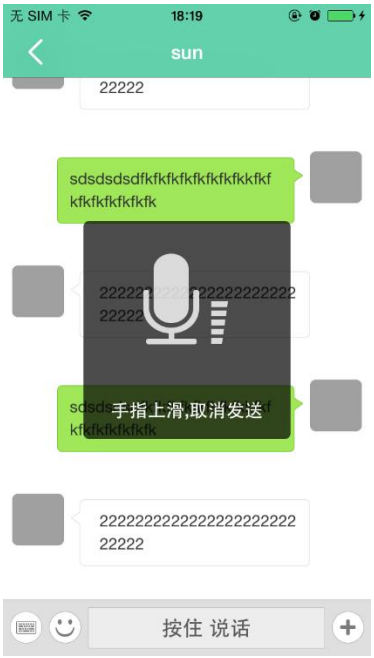
5.1 用户界面设计

给出系统用户界面的总体设计决策，和典型的用户界面风格。

说明：如果系统提供了对其他系统的接口，如从其他软件系统导入、导出数据，必须在此说明。

图为用户界面友好的聊天界面，支持用户输入模块(包含输入框、录音、表情选择、拍照、从相册选取照片功能)、录音模块。用户可以通过界面发送文字、语音、图片等，进行使用自动问答系统的提问功能和聊天功能。

具体的地址为：http://pan.baidu.com/s/1mg927o8



5.2 外部接口设计.

5.2.1 硬件接口

阿里云服务器、Android 手机

5.2.2 软件接口

Mysql、Redis、Neo4J、Apache Tomcat、Linux 服务器

5.3 内部接口设计

对系统各构件（模块）之间的接口进行说明。

本系统主要使用自然语言处理技术，其中主要包括分词技术（这里使用 N-最短路劲分词以及 CRF 条件随机场分词技术），词典技术(这里使用通过分词技术而来的极速词典技术以及自定义的词典技术)，人名地名的识别技术，关键词提取技术、自动摘要提取技术、短语提取技术、拼音转换技术。下面是代码中的接口实现。

5.3.1 N-最短路径分词

```
    Segment nShortSegment = new
NShortSegment().enableCustomDictionary(false).enablePlaceRecognize(true).en
ableOrganizationRecognize(true);
    Segment shortestSegment = new
DijkstraSegment().enableCustomDictionary(false).enablePlaceRecognize(true).
enableOrganizationRecognize(true);
    String[] testCase = new String[]{
            "今天，刘志军案的关键人物,山西女商人丁书苗在市二中院出庭受审。",
            "刘喜杰石国祥会见吴亚琴先进事迹报告团成员",
            };
    for (String sentence : testCase)
    {
        System.out.println("N-最短分词：" + nShortSegment.seg(sentence) + "\n
最短路分词：" + shortestSegment.seg(sentence));
    }
```

5.3.2 CRF 分词

```
    Segment segment = new CRFSegment();
    segment.enablePartOfSpeechTagging(true);
    List<Term> termList = segment.seg("你看过穆赫兰道吗");
    System.out.println(termList);
    for (Term term : termList)
    {
        if (term.nature == null)
        {
            System.out.println("识别到新词：" + term.word);
        }
    }
```

5.3.3 极速词典分词

```
    /**
     * 演示极速分词，基于 AhoCorasickDoubleArrayTrie 实现的词典分词，适用于“高吞吐
量”“精度一般”的场合
     */
    public class DemoHighSpeedSegment
    {
        public static void main(String[] args)
        {
            String text = "江西鄱阳湖干枯，中国最大淡水湖变成大草原";
```

```
            System.out.println(SpeedTokenizer.segment(text));
            long start = System.currentTimeMillis();
            int pressure = 1000000;
            for (int i = 0; i < pressure; ++i)
            {
                SpeedTokenizer.segment(text);
            }
            double costTime = (System.currentTimeMillis() - start) /
(double)1000;
            System.out.printf("分词速度：%.2f 字每秒", text.length() * pressure
/ costTime);
        }
    }
```

5.3.4 用户自定义词典

```
    /**
     * 演示用户词典的动态增删
     */
    public class DemoCustomDictionary
    {
        public static void main(String[] args)
        {
            // 动态增加
            CustomDictionary.add("攻城狮");
            // 强行插入
            CustomDictionary.insert("白富美", "nz 1024");
            // 删除词语（注释掉试试）
    //        CustomDictionary.remove("攻城狮");
            System.out.println(CustomDictionary.add("单身狗", "nz 1024 n 1"));
            System.out.println(CustomDictionary.get("单身狗"));

            String text = "攻城狮逆袭单身狗，迎娶白富美，走上人生巅峰";  // 怎么可
能噗哈哈！

            // AhoCorasickDoubleArrayTrie 自动机分词
            final char[] charArray = text.toCharArray();
            CustomDictionary.parseText(charArray, new
AhoCorasickDoubleArrayTrie.IHit<CoreDictionary.Attribute>()
            {
                @Override
                public void hit(int begin, int end, CoreDictionary.Attribute
value)
                {
```

```
                System.out.printf("[%d:%d]=%s %s\n", begin, end, new
String(charArray, begin, end - begin), value);
            }
        });
        // trie 树分词
        BaseSearcher searcher = CustomDictionary.getSearcher(text);
        Map.Entry entry;
        while ((entry = searcher.next()) != null)
        {
            System.out.println(entry);
        }

        // 标准分词
        System.out.println(HanLP.segment(text));
    }
}
```

5.3.5 中国人名识别

```
    String[] testCase = new String[]{
            "签约仪式前，秦光荣、李纪恒、仇和等一同会见了参加签约的企业家。",
            "王国强、高峰、汪洋、张朝阳光着头、韩寒、小四",
            "张浩和胡健康复员回家了",
            "王总和小丽结婚了",
            "编剧邵钧林和稽道青说",
            "这里有关天培的有关事迹",
            "龚学平等领导,邓颖超生前",
            };
    Segment segment = HanLP.newSegment().enableNameRecognize(true);
    for (String sentence : testCase)
    {
        List<Term> termList = segment.seg(sentence);
        System.out.println(termList);
    }
```

5.3.6 音译人名识别

```
    String[] testCase = new String[]{
                    "一桶冰水当头倒下，微软的比尔盖茨、Facebook 的扎克伯格跟桑德博
格、亚马逊的贝索斯、苹果的库克全都不惜湿身入镜，这些硅谷的科技人，飞蛾扑火似地牺牲演
出，其实全为了慈善。",
                    "世界上最长的姓名是简森·乔伊·亚历山大·比基·卡利斯勒·达夫·埃利
奥特·福克斯·伊维鲁莫·马尔尼·梅尔斯·帕特森·汤普森·华莱士·普雷斯顿。",
            };
```

```
    Segment segment = HanLP.newSegment().enableTranslatedNameRecognize(true);
    for (String sentence : testCase)
    {
        List<Term> termList = segment.seg(sentence);
        System.out.println(termList);
    }
```

5.3.7 地名识别

```
    String[] testCase = new String[]{
            "武胜县新学乡政府大楼门前锣鼓喧天",
            "蓝翔给宁夏固原市彭阳县红河镇黑牛沟村捐赠了挖掘机",
    };
    Segment segment = HanLP.newSegment().enablePlaceRecognize(true);
    for (String sentence : testCase)
    {
        List<Term> termList = segment.seg(sentence);
        System.out.println(termList);
    }
```

5.3.8 机构名识别

```
    String[] testCase = new String[]{
        "我在上海林原科技有限公司兼职工作，",
        "我经常在台川喜宴餐厅吃饭，",
        "偶尔去地中海影城看电影。",
    };
    Segment segment = HanLP.newSegment().enableOrganizationRecognize(true);
    for (String sentence : testCase)
    {
        List<Term> termList = segment.seg(sentence);
        System.out.println(termList);
    }
```

5.3.9 关键词提取

```
    String content = "程序员(英文 Programmer)是从事程序开发、维护的专业人员。一般将程序员分为程序设计人员和程序编码人员，但两者的界限并不非常清楚，特别是在中国。软件从业人员分为初级程序员、高级程序员、系统分析员和项目经理四大类。";
    List<String> keywordList = HanLP.extractKeyword(content, 5);
    System.out.println(keywordList);
```

5.3.10 自动摘要

```
    String document = "算法可大致分为基本算法、数据结构的算法、数论算法、计算几何的算法、图的算法、动态规划以及数值分析、加密算法、排序算法、检索算法、随机化算法、并行算法、厄米变形模型、随机森林算法。\n" +
            "算法可以宽泛的分为三类，\n" +
            "一，有限的确定性算法，这类算法在有限的一段时间内终止。他们可能要花很长时间来执行指定的任务，但仍将在一定的时间内终止。这类算法得出的结果常取决于输入值。\n" +
            "二，有限的非确定算法，这类算法在有限的时间内终止。然而，对于一个（或一些）给定的数值，算法的结果并不是唯一的或确定的。\n" +
            "三，无限的算法，是那些由于没有定义终止定义条件，或定义的条件无法由输入的数据满足而不终止运行的算法。通常，无限算法的产生是由于未能确定的定义终止条件。";
    List<String> sentenceList = HanLP.extractSummary(document, 3);
    System.out.println(sentenceList);
```

5.3.11 短语提取

```
    String text = "算法工程师\n" +
                    "算法（Algorithm）是一系列解决问题的清晰指令，也就是说，能够对一定规范的输入，在有限时间内获得所要求的输出。如果一个算法有缺陷，或不适合于某个问题，执行这个算法将不会解决这个问题。不同的算法可能用不同的时间、空间或效率来完成同样的任务。一个算法的优劣可以用空间复杂度与时间复杂度来衡量。算法工程师就是利用算法处理事物的人。\n" +
                    "\n" +
                    "1 职位简介\n" +
                    "算法工程师是一个非常高端的职位；\n" +
                    "专业要求：计算机、电子、通信、数学等相关专业；\n" +
                    "学历要求：本科及其以上的学历，大多数是硕士学历及其以上；\n" +
                    "语言要求：英语要求是熟练，基本上能阅读国外专业书刊；\n" +
                    "必须掌握计算机相关知识，熟练使用仿真工具 MATLAB 等，必须会一门编程语言。\n" +
                    "\n" +
                    "2 研究方向\n" +
                    "视频算法工程师、图像处理算法工程师、音频算法工程师 通信基带算法工程师\n" +
                    "\n" +
                    "3 目前国内外状况\n" +
                    "目前国内从事算法研究的工程师不少，但是高级算法工程师却很少，是一个非常紧缺的专业工程师。算法工程师根据研究领域来分主要有音频/视频算法处理、图像技术方面的二维信息算法处理和通信物理层、雷达信号处理、生物医学信号处理等领域的一维信息算法处理。\n" +
```

```
                "在计算机音视频和图形图像技术等二维信息算法处理方面目前比较先进的视频处理算法：机器视觉成为此类算法研究的核心；另外还有 2D 转 3D 算法(2D-to-3D conversion)，去隔行算法(de-interlacing)，运动估计运动补偿算法(Motion estimation/Motion Compensation)，去噪算法(Noise Reduction)，缩放算法(scaling)，锐化处理算法(Sharpness)，超分辨率算法(Super Resolution),手势识别(gesture recognition),人脸识别(face recognition)。\n" +
                "在通信物理层等一维信息领域目前常用的算法：无线领域的 RRM、RTT，传送领域的调制解调、信道均衡、信号检测、网络优化、信号分解等。\n" +
                "另外数据挖掘、互联网搜索算法也成为当今的热门方向。\n" +
                "算法工程师逐渐往人工智能方向发展。";
    List<String> phraseList = HanLP.extractPhrase(text, 10);
    System.out.println(phraseList);
```

5.3.12 拼音转换

```
    /**
     * 汉字转拼音
     */
    public class DemoPinyin
    {
        public static void main(String[] args)
        {
            String text = "重载不是重任";
            List<Pinyin> pinyinList = HanLP.convertToPinyinList(text);
            System.out.print("原文,");
            for (char c : text.toCharArray())
            {
                System.out.printf("%c,", c);
            }
            System.out.println();

            System.out.print("拼音（数字音调）,");
            for (Pinyin pinyin : pinyinList)
            {
                System.out.printf("%s,", pinyin);
            }
            System.out.println();

            System.out.print("拼音（符号音调）,");
            for (Pinyin pinyin : pinyinList)
            {
                System.out.printf("%s,", pinyin.getPinyinWithToneMark());
            }
            System.out.println();
```

```
        System.out.print("拼音（无音调）,");
        for (Pinyin pinyin : pinyinList)
        {
            System.out.printf("%s,", pinyin.getPinyinWithoutTone());
        }
        System.out.println();

        System.out.print("声调,");
        for (Pinyin pinyin : pinyinList)
        {
            System.out.printf("%s,", pinyin.getTone());
        }
        System.out.println();

        System.out.print("声母,");
        for (Pinyin pinyin : pinyinList)
        {
            System.out.printf("%s,", pinyin.getShengmu());
        }
        System.out.println();

        System.out.print("韵母,");
        for (Pinyin pinyin : pinyinList)
        {
            System.out.printf("%s,", pinyin.getYunmu());
        }
        System.out.println();

        System.out.print("输入法头,");
        for (Pinyin pinyin : pinyinList)
        {
            System.out.printf("%s,", pinyin.getHead());
        }
        System.out.println();
    }
}
```

6. 数据结构设计

6.1 公共数据结构设计

Trie 树（字典树）为前缀树，是为了支持泛型、遍历、储存、载入，为自然语言处理中最常用的高效检索数据结构。

6.2 数据库设计

6.2.1 文档数据库

主要使用文档进行初步存储。其中词典有两个形态：文本文件(filename.txt)和缓存文件(filename.txt.bin 或 filename.txt.trie.dat 和 filename.txt.trie.value)。

- 文本文件
  - 采用明文储存，UTF-8 编码，CRLF 换行符。
- 缓存文件
  - 就是一些二进制文件，通常在文本文件的文件名后面加上.bin 表示。有时候是.trie.dat 和.trie.value。后者是历史遗留产物，分别代表 trie 树的数组和值。
  - 修改了任何词典，只有删除缓存才能生效。

词典说明：

词典分为词频词性词典和词频词典。

- 词频词性词典
  - 每一行代表一个单词，格式遵从[单词] [词性 A] [A 的频次] [词性 B] [B 的频次] 。
- 词频词典
  - 每一行代表一个单词，格式遵从[单词] [单词的频次]。
  - 每一行的分隔符为空格符或制表符

少数词典有自己的专用格式，比如同义词词典兼容《同义词词林扩展版》的文本格式，而转移矩阵词典则是一个 csv 表格。

6.2.2 redis 数据库表的设计

参考（https://upload.wikimedia.org/wikipedia/commons/f/f7/MediaWiki_1.24.1_database_schema.svg）

数据库的表结构	表名称	设计者	审核者	完成日期

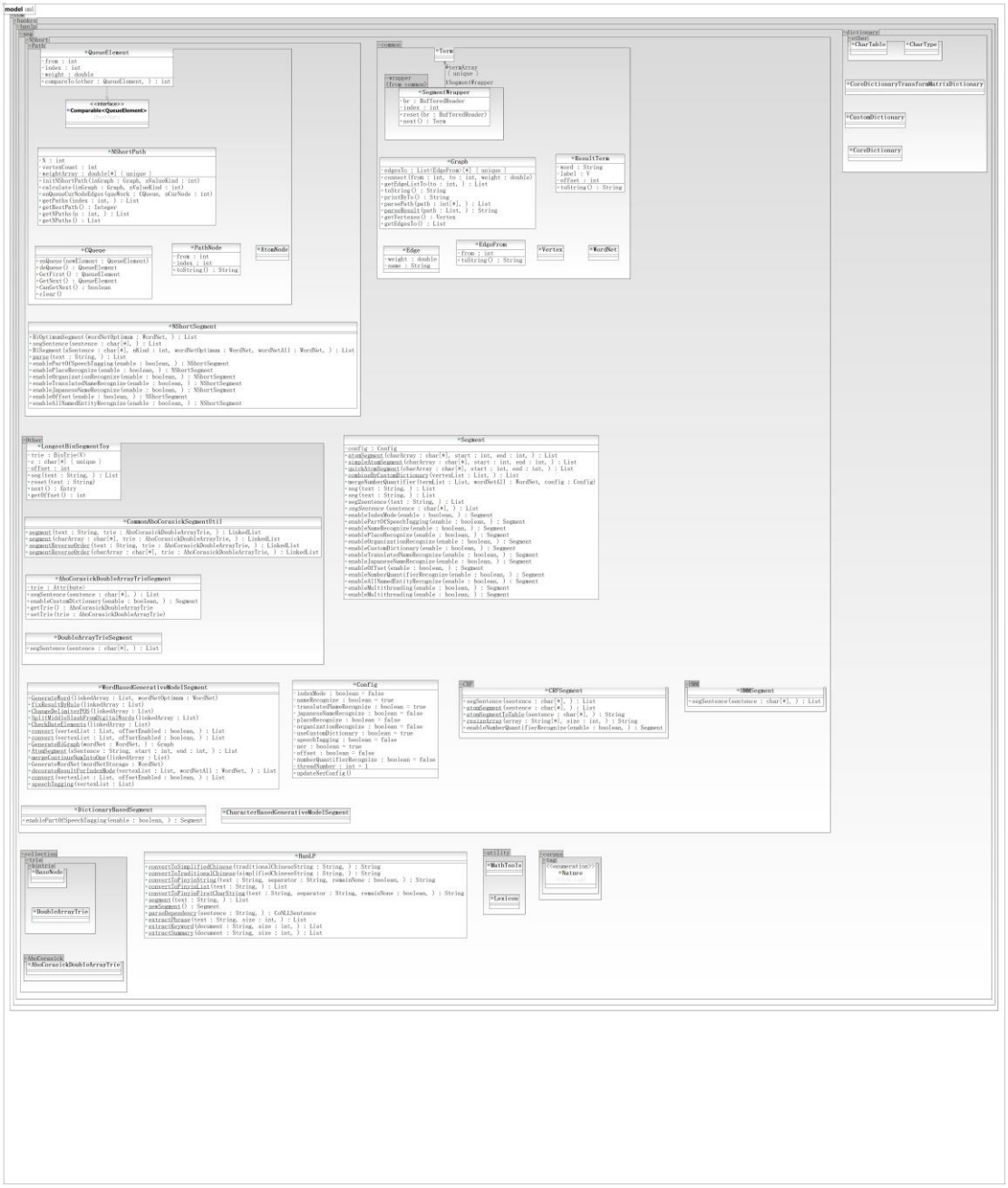
字段代号	名称	类型	值域	数据项名	索引或键	备注	缩写词

7. 详细设计

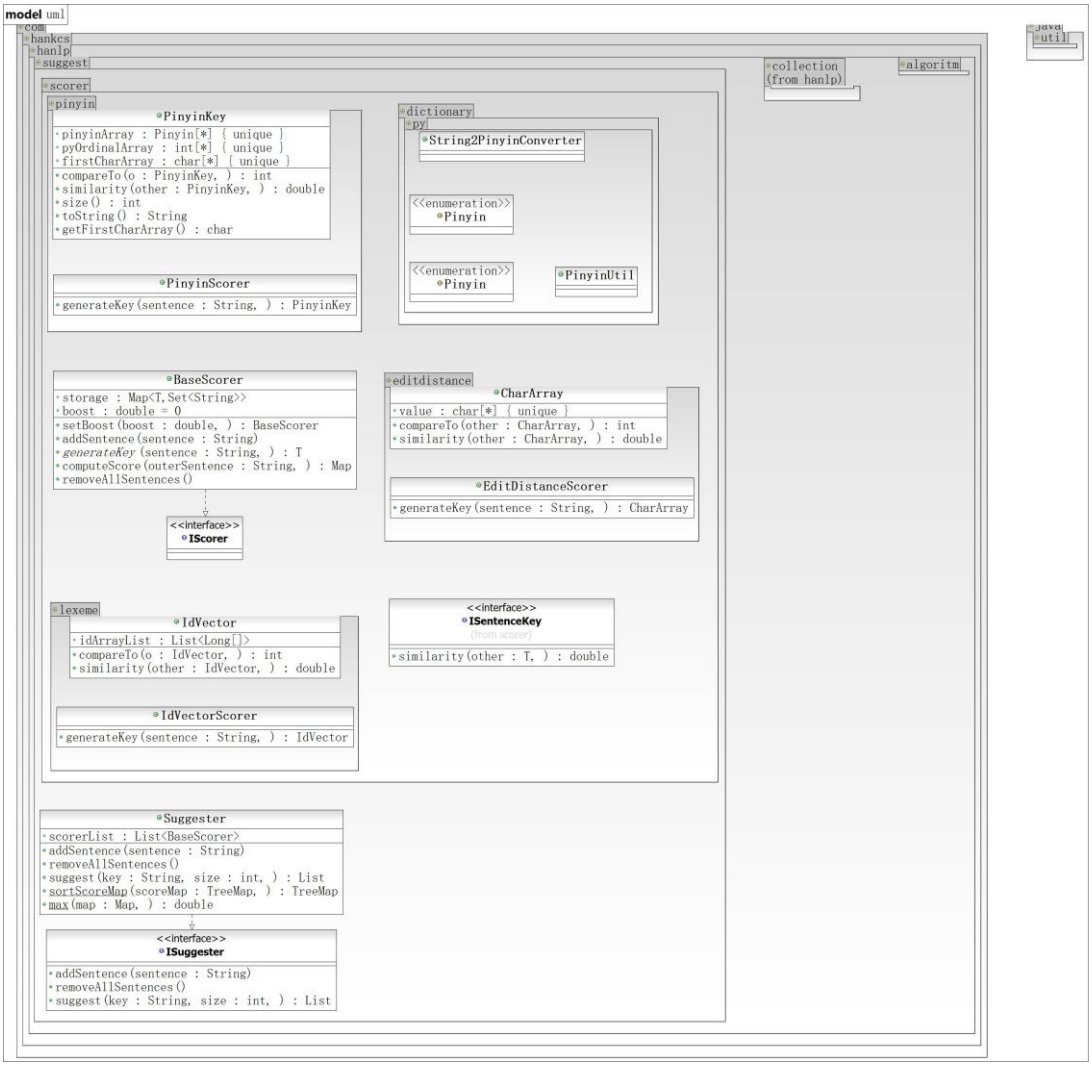
每一小节给出一个模块（构件）的详细设计方案。包括模块概述、模块的接口说明（即输入、输出）、以及内部结构设计。其中内部结构又可以考虑从静态、动态结构两个方面阐述；静态结构应给出该模块（构件）的类结构（类图），动态结构应给出该模块关键业务流程的交互模型（顺序图），还可根据实际情况给出状态图（某个构件或对象的状态迁移）和活动图（某个算法的实现流程）等内容。

（详见 team_16 项目目录，此图为 uml2）

7.1.1 分词模块

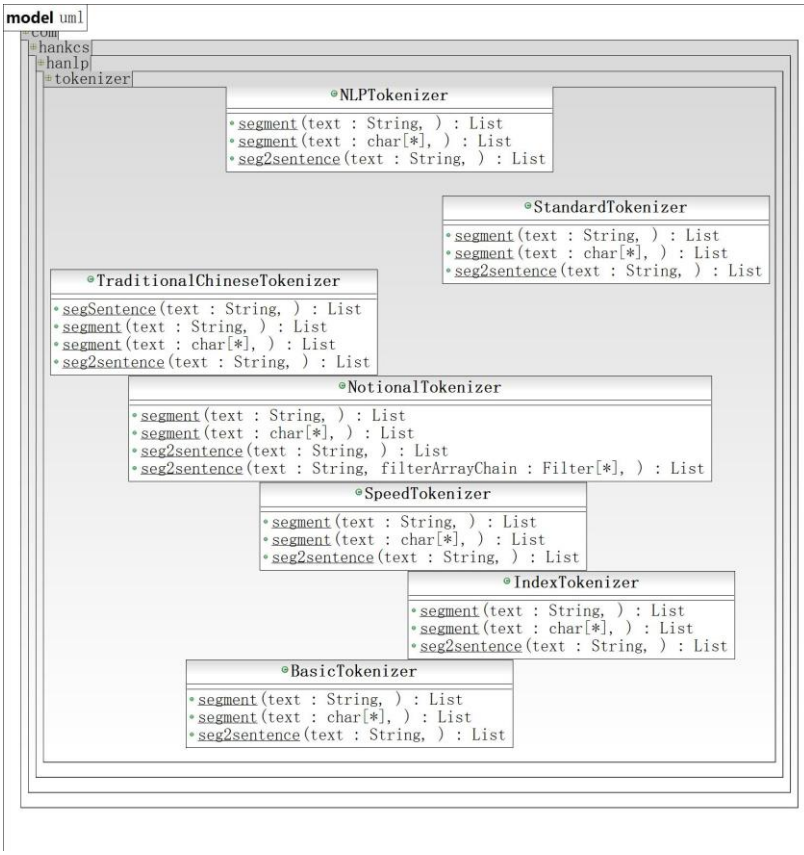


7.1.2 文本推荐模板

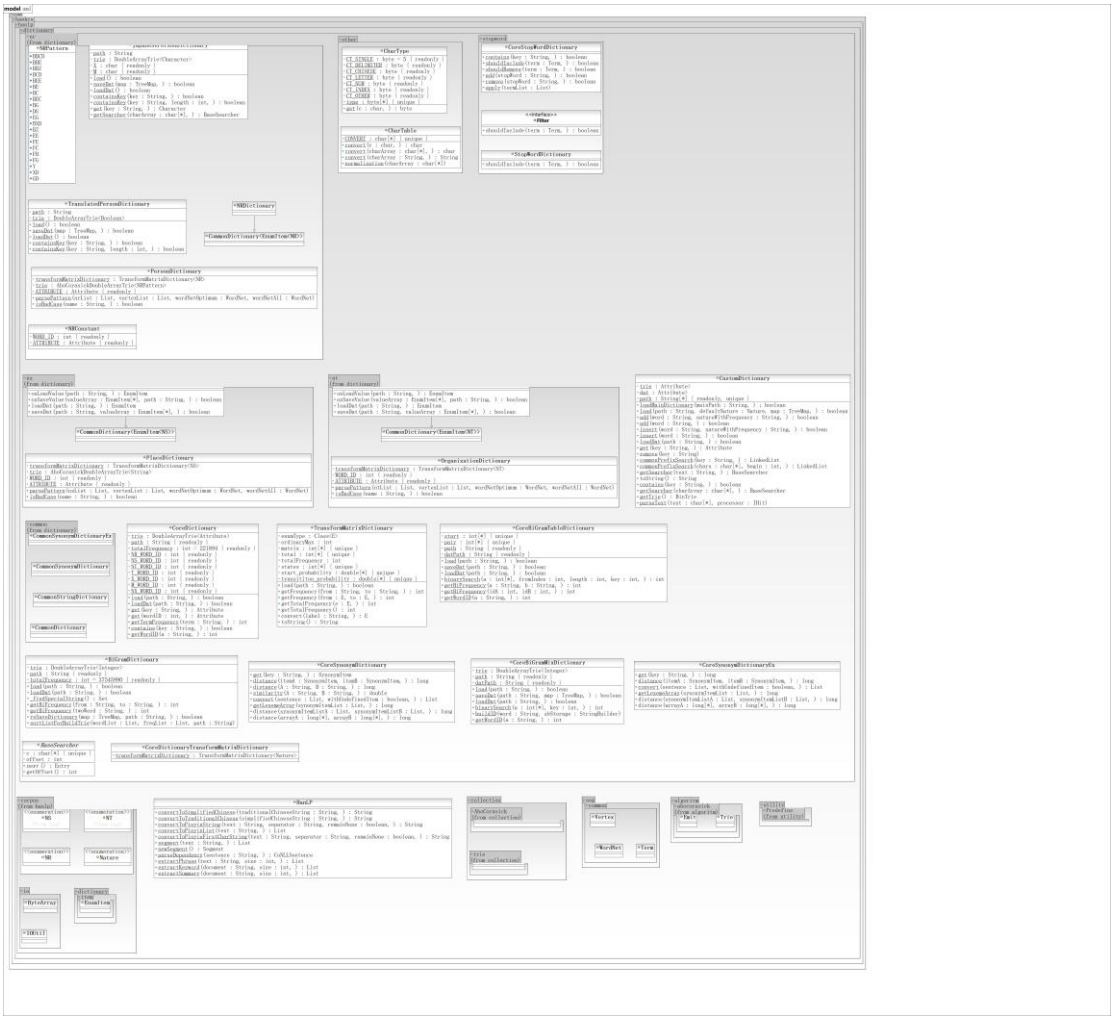

model uml
com
hankcs
hanlp
suggest
scorer
pinyin
PinyinKey
pinyinArray : Pinyin[*] { unique }
pyOrdinalArray : int[*] { unique }
firstCharArray : char[*] { unique }
compareTo(o : PinyinKey,) : int
similarity(other : PinyinKey,) : double
size() : int
toString() : String
getFirstCharArray() : char
PinyinScorer
generateKey(sentence : String,) : PinyinKey
dictionary
py
String2PinyinConverter
<<enumeration>>
Pinyin
<<enumeration>>
Pinyin
PinyinUtil
collection
(from hanlp)
algoritm
java
util
BaseScorer
storage : Map<T,Set<String>>
boost : double = 0
setBoost(boost : double,) : BaseScorer
addSentence(sentence : String)
generateKey(sentence : String,) : T
computeScore(outerSentence : String,) : Map
removeAllSentences()
<<interface>>
IScorer
editdistance
CharArray
value : char[*] { unique }
compareTo(other : CharArray,) : int
similarity(other : CharArray,) : double
EditDistanceScorer
generateKey(sentence : String,) : CharArray
lexeme
IdVector
idArrayList : List<Long[]>
compareTo(o : IdVector,) : int
similarity(other : IdVector,) : double
IdVectorScorer
generateKey(sentence : String,) : IdVector
<<interface>>
ISentenceKey
(from scorer)
similarity(other : T,) : double
Suggester
scorerList : List<BaseScorer>
addSentence(sentence : String)
removeAllSentences()
suggest(key : String, size : int,) : List
sortScoreMap(scoreMap : TreeMap,) : TreeMap
max(map : Map,) : double
<<interface>>
ISuggester
addSentence(sentence : String)
removeAllSentences()
suggest(key : String, size : int,) : List

7.1.3 抽取概要模板

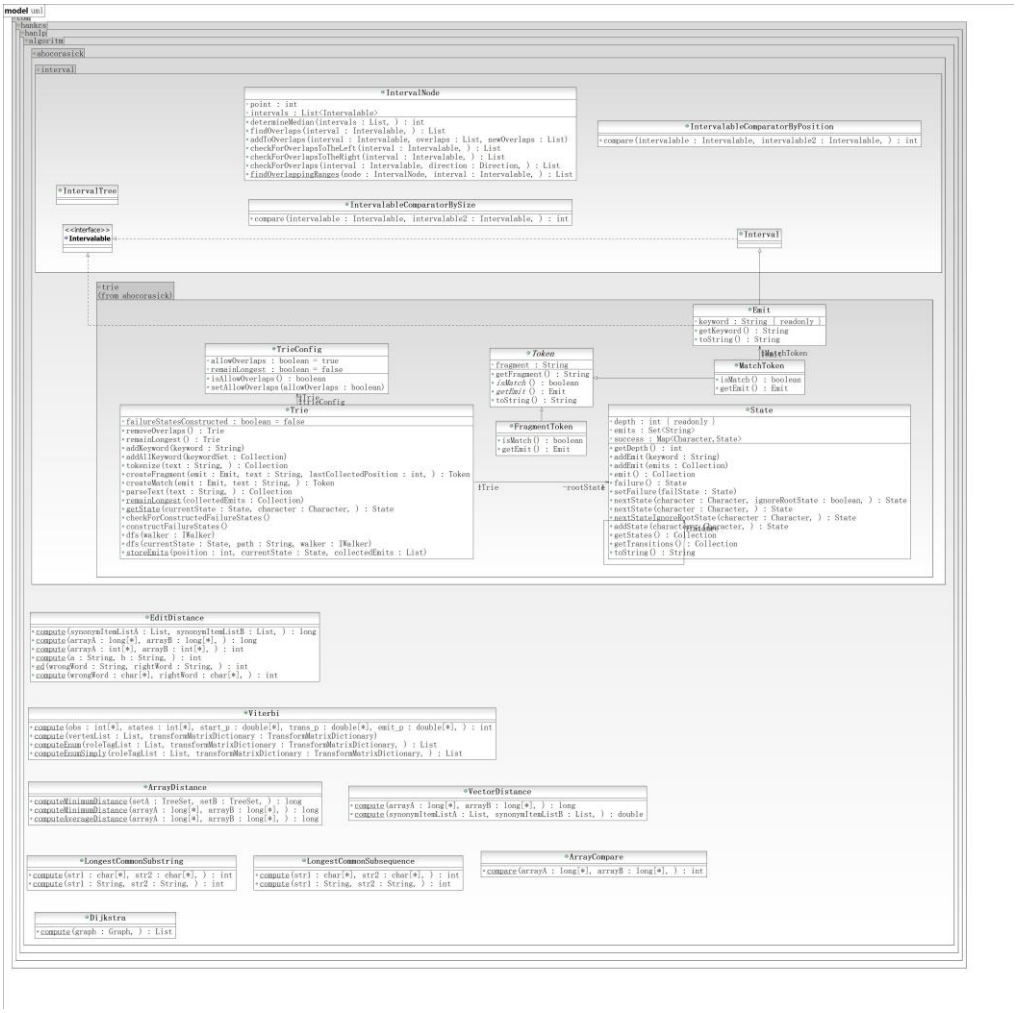
7.1.4 抽取标志符模板



7.1.5 词典模板



7.1.6 基础算法模板


model uml
com
banker
huolig
algorithm
ahocorasick
interval
IntervalNode
point : int
intervals : List<Intervalable>
determineMedian(intervals : List,) : int
findOverlaps(interval : Intervalable,) : List
addToOverlaps(interval : Intervalable, overlaps : List, newOverlaps : List)
checkForOverlapsToTheLeft(interval : Intervalable,) : List
checkForOverlapsToTheRight(interval : Intervalable,) : List
checkForOverlaps(interval : Intervalable, direction : Direction,) : List
findOverlappingRanges(node : IntervalNode, interval : Intervalable,) : List
IntervalTree
IntervalableComparatorBySize
compare(intervalable : Intervalable, intervalable2 : Intervalable,) : int
IntervalableComparatorByPosition
compare(intervalable : Intervalable, intervalable2 : Intervalable,) : int
<<interface>>
Intervalable
Interval
trie
(from ahocorasick)
Emit
keyword : String { readonly }
getKeyword() : String
toString() : String
IMatchToken
MatchToken
isMatch() : boolean
getEmit() : Emit
TrieConfig
allowOverlaps : boolean = true
remainLongest : boolean = false
isAllowOverlaps() : boolean
setAllowOverlaps(allowOverlaps : boolean)
ITrieConfig
Token
fragment : String
getFragment() : String
isMatch() : boolean
getEmit() : Emit
toString() : String
Trie
failureStatesConstructed : boolean = false
removeOverlaps() : Trie
remainLongest() : Trie
addKeyword(keyword : String)
addAllKeyword(keywordSet : Collection)
tokenize(text : String,) : Collection
createFragment(emit : Emit, text : String, lastCollectedPosition : int,) : Token
createMatch(emit : Emit, text : String,) : Token
parseText(text : String,) : Collection
remainLongest(collectedEmits : Collection)
getState(currentState : State, character : Character,) : State
checkForConstructedFailureStates()
constructFailureStates()
dfs(walker : IWalker)
dfs(currentState : State, path : String, walker : IWalker)
storeEmits(position : int, currentState : State, collectedEmits : List)
FragmentToken
isMatch() : boolean
getEmit() : Emit
ITrie
rootState
State
depth : int { readonly }
emits : Set<String>
success : Map<Character,State>
getDepth() : int
addEmit(keyword : String)
addEmit(emits : Collection)
emit() : Collection
failure() : State
setFailure(failState : State)
nextState(character : Character, ignoreRootState : boolean,) : State
nextState(character : Character,) : State
nextStateIgnoreRootState(character : Character,) : State
addState(character : Character,) : State
getStates() : Collection
getTransitions() : Collection
toString() : String
EditDistance
compute(synonymItemListA : List, synonymItemListB : List,) : long
compute(arrayA : long[*], arrayB : long[*],) : long
compute(arrayA : int[*], arrayB : int[*],) : int
compute(a : String, b : String,) : int
ed(wrongWord : String, rightWord : String,) : int
compute(wrongWord : char[*], rightWord : char[*],) : int
Viterbi
compute(obs : int[*], states : int[*], start_p : double[*], trans_p : double[*], emit_p : double[*],) : int
compute(vertexList : List, transformMatrixDictionary : TransformMatrixDictionary)
computeEnum(roleTagList : List, transformMatrixDictionary : TransformMatrixDictionary,) : List
computeEnumSimply(roleTagList : List, transformMatrixDictionary : TransformMatrixDictionary,) : List
ArrayDistance
computeMinimumDistance(setA : TreeSet, setB : TreeSet,) : long
computeMinimumDistance(arrayA : long[*], arrayB : long[*],) : long
computeAverageDistance(arrayA : long[*], arrayB : long[*],) : long
VectorDistance
compute(arrayA : long[*], arrayB : long[*],) : long
compute(synonymItemListA : List, synonymItemListB : List,) : double
LongestCommonSubstring
compute(str1 : char[*], str2 : char[*],) : int
compute(str1 : String, str2 : String,) : int
LongestCommonSubsequence
compute(str1 : char[*], str2 : char[*],) : int
compute(str1 : String, str2 : String,) : int
ArrayCompare
compare(arrayA : long[*], arrayB : long[*],) : int
Dijkstra
compute(graph : Graph,) : List

7.1.7 数据结构模板