华院数据中文地址魔方大赛

结合地址树推理和正则匹配的中文地址识别

技术方案说明文档

团队名称：甲基苯丙胺

一、 技术选型

1) 概述

a) 应用背景

中文地址蕴含了丰富信息，对电子商务应用挖掘具有很高的潜在价值。但长期以来，由于中文自然语言处理的复杂性、地址命名的不规范性，导致地址信息不能被充分分析和挖掘。通过对地址信息的标准化处理，使得基于地址的多维度量化挖掘分析成为可能，为不同场景

模式下的电子商务应用挖掘提供了更加丰富的方法和手段，因此具有重要的现实意义

b) 业务目的

中文地址在统计上呈现一定的规律性。本算法系统致力于设计地址信息的标准化算法，在充分的统计分析基础上，设计合理的推理模型和模式匹配方法，实现省、市、区等 10 个地址级别字段的自动识别。

c) 技术约束

本算法系统适用于国内中文地址的自动识别。目前实现的功能模块能够识别中文地址魔方大赛的 50 万条数据，并在 B 榜排名获得 top1 的成绩。

通过增加国外行政区划数据库，算法系统可以得到扩展，解决国外地址的识别问题。

本文的算法系统框架具有普适性。在一定的统计分析基础上，可以实现任意地址的高精度识别。

2) 方案说明

a) 基于行政规划地名的识别方案

i. 前提假设

中文地址中的省市县等属于国家行政区划，名称固定且不经常发生变化，可以将其作为标准数据库，与待分析的中文地址进行比较分析。

ii. 技术原理

利用中文分词中的正向最长匹配策略，可以快速切分出数据库存在的中文地址。考虑到分级地址是一个树形结构，可以建立这颗地址树并在上面进行推理，实现中文地址的识别。

b) 基于特定规则的模式匹配识别方案

i. 前提假设

由于中文地址随等级细分呈指数级增长，更精细的中文地址难以建立标准化的地名数据库。但通过一定的分析可以发现，每一级中文地址都存在一些共有的特征模式。如果能分析出所有模式，就能解决中文地址识别的问题。

ii. 技术原理

通过对特定的地址识别任务，设计精巧的规则，就能以很高的精度完成给定的识别任务。实现方法可以通过构造一系列正则表达式，按照地址等级递增的顺序完成识别任务。

c) 基于统计学习的识别方案

i. 前提假设

中文地址千变万化，考虑到语言的随机性，很难通过手工的方式予以穷举而形成一种形式化的规则库。但是大量的中文地址在统计上呈现一定的规律性。在拥有足够多的观测样例的基础上，可以利用统计学习（机器学习）的方式对中文地址进行分析。

HMM 模型提供了语法规则的概率存在和穷举局限性问题。通过对训练语料的学习，HMM 模型可获取这种语法规则知识进而很好的支持后续的文本处理。

ii. 技术原理

一种机器学习系统的整体流程如下图所示。整个系统从纵向上可看作是两层 HMM 的相继使用，底层的输入观察序列集合是文本粗切分结果，输出则是粗分文本的最佳词性标注序列；上层模型以底层模型为基础，利用最佳词性标注序列作为输入观察序列，输出则是文本的最佳边界标记序列。

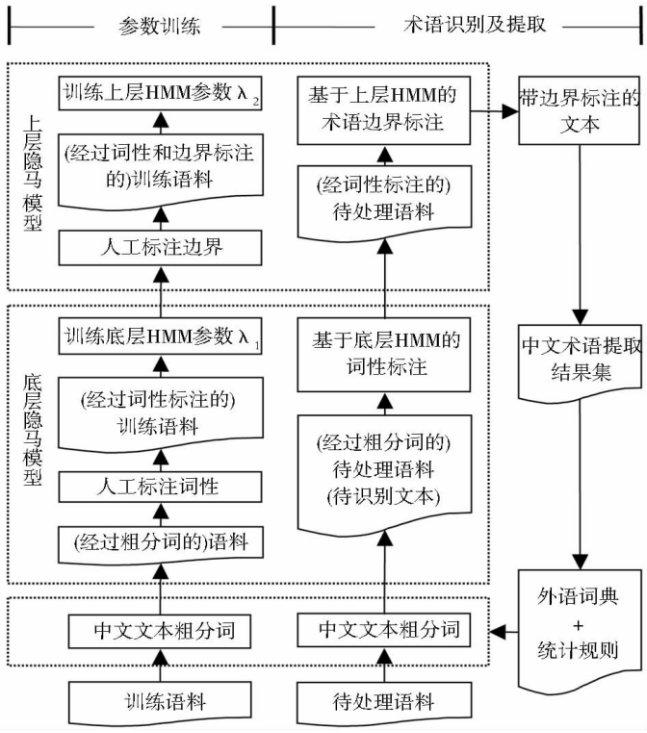


图1 基于双层隐马模型的中文术语提取系统的总体流程

d) 对比分析

考虑本次大赛提供的中文地址训练集数据只有 500 条，对于机器学习模型而言，少量的数据和中文语义鸿沟之间存在巨大的矛盾，基于机器学习的方法面临巨大的挑战。

对于基于行政区划的方案，可以很好的实现前 3 级的识别任务。对于更精细的地址分级，则可以利用基于规则的方式实现识别。经过对训练集数据的简单分析，发现 10 级地址的识别结果存在比较明显的规律性。因此，采用结合行政区划地名和规则的策略实现中文地址识别，符合各自方法的前提假设，是比较可取的策略。

3) 方案选择

综合考虑前提假设的合理性和算法实现难度，本文选择行政区划地名和特定规则模式匹配的方案实现算法系统。

二、 架构设计

1) 系统架构

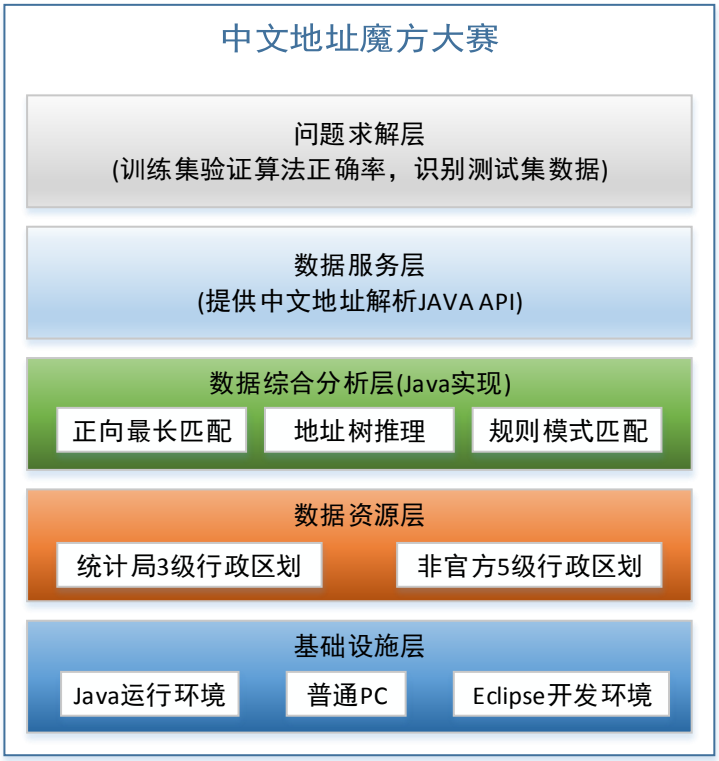


图2 系统架构图

2) 架构优势

选取此架构和流程有一下几点优势：

a) Java 代码可跨平台执行，并且算法系统对外提供一个 API，调用简明方便。

b) 算法依赖最新的行政区划数据，能够对不规范的地名进行纠正。当行政区划有更新变动时，只需替换数据，无须重构代码。

c) 基于规则的模式匹配，使用正则表达式表示规则。规则的表示简明，并且在实际应用中效率很高。一般而言，考虑到中文表达的习惯，每一级地址所固有的模式不会改变，因此特定的一组规则适用性很广。若发现新的模式，修正规则库即可。

d) 算法模块化，扩展方便。对于已有行政区划的地名识别，本系统的精确率已经接近100%。而对于未登记的地名，可以通过增加算法模块的方式继续改进。对于其他国家地名识别任务，也可以通过修改行政区划数据库和规则数据库实现。在该架构下扩展系统很方便。

三、 模块划分

1) 系统流程

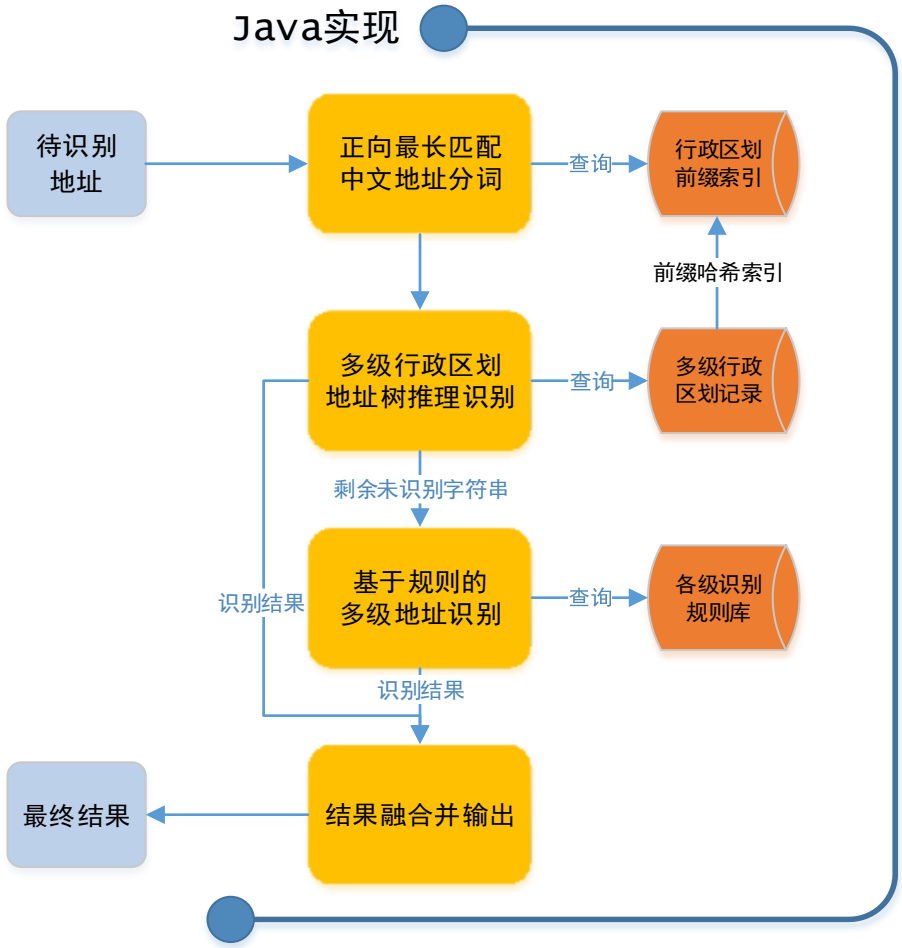


图3 系统流程图

算法系统的流程如图 3 所示。系统依赖已知的地名数据和匹配规则。

系统首先对源文本进行正向最长匹配分词，分词结果由地址树推理分析，得到精确的多级地名识别结果。这一步的正确率很高，因此有理由将识别出的地名从源文本中删去，并将剩余文本传递给下一模块处理。

基于规则的多级地址识别算法，依赖于对每一级地名的分析，从数据中发现最一般的规律，从而设计合理且有效的识别规则。规则采用正则表达式描述，简明快速且不失一般性。此种方法可以识别未在数据库中出现的地名，泛化性能很好。此模块接收到地址树传递的输入后，自动分析出更复杂更一般的多级地址字段。

算法系统最终将地址树的推理结果和规则匹配的识别结果融合，作为算法系统的最终输出。

整套算法系统使用 Java 语言实现，对外提供接口 AddressParser String[] parse(String str)，

方便调用。整套系统性能优异，在效率上每次查询只需 37us，在精确度上 B 榜最终排名 top1，综合分数为 4755.6 分。

2) 正向最长匹配和地址树推理模块

a) 接口 CityMatch String[] search(String str)

输入：String str，源地址字符串

输出：String ans[]. ans[0]为去除识别出的实体后剩余的字符串。ans[1-10]保存识别结果。

b) 正向最长匹配算法模块

算法思路：

1. 利用 3 级行政区划数据，构建地名的前缀哈希表和地址树。
2. 置 i=0。
3. 对输入字符串 S，从 S 的第 i 个字符开始，搜索在前缀哈希表中出现过的以该字符起始的最长子串。若长度大于一定阈值，则认为匹配成功，识别该地名；若不能识别，i←i+1，转 5。
4. 将该地名在地址树加上分数 $X/2^{count}$，X 可以是任意数，不妨取 2^{20}，count 已是识别地名的总数。i←识别子串的下一个位置。
5. 若 i 小于 S 的长度，重复 3。否则结束。

流程图如下：

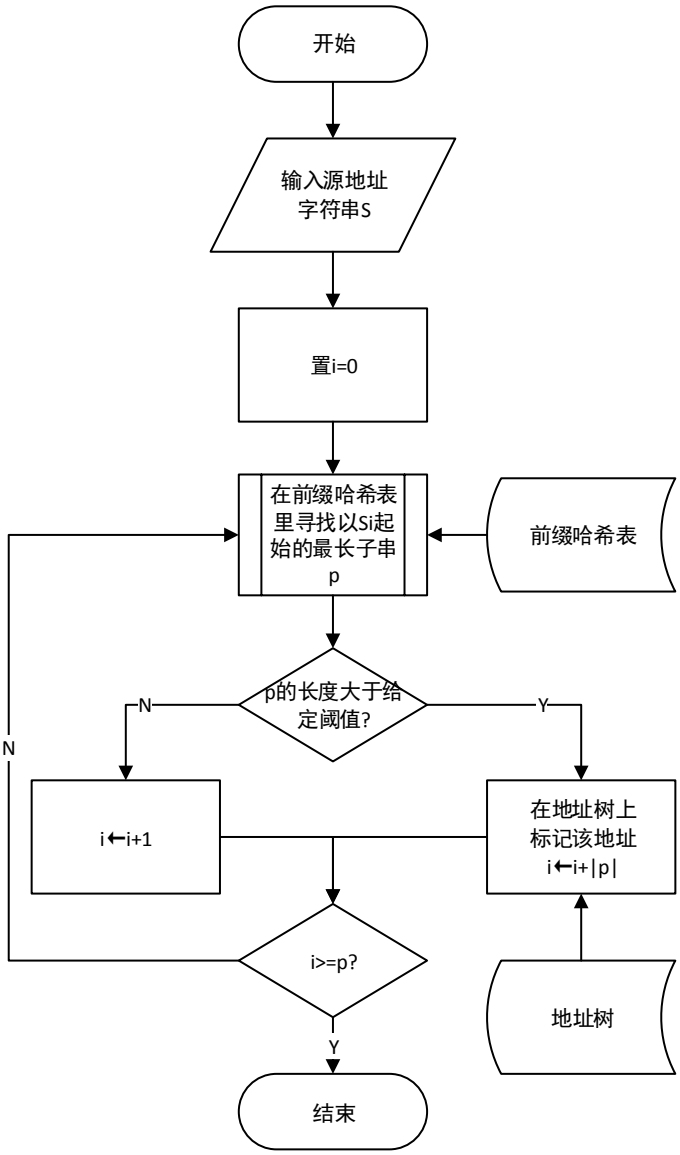


图4　正向最长匹配算法流程图

c)　地址树推理模块

算法思路：

枚举地址树上被加分的节点，计算其到树根的分数，显然经过的节点就是多级地址。取分数最高的节点作为答案，就可以得到最终的推理结果。

举例说明：南京市玄武区市政府

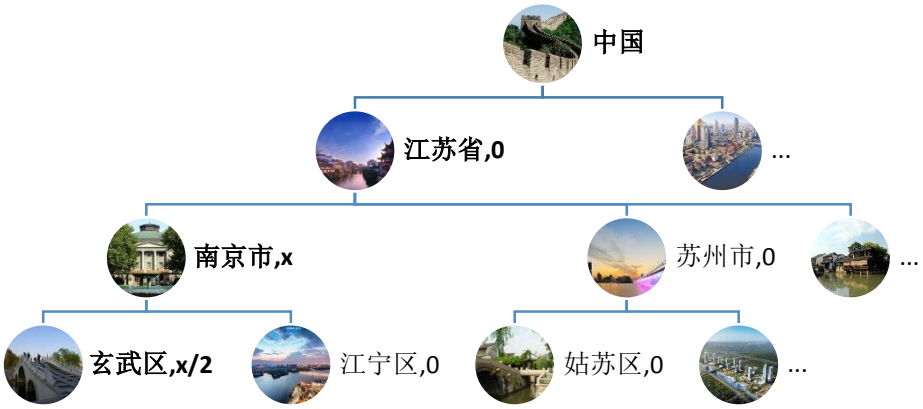


图5　地址树推理示意图

有分数变动的节点是“南京市”和“玄武区”节点，其分数分别为 x 和 1.5x。因此选择玄武区这个节点作为识别结果。其到树根经过的点为 玄武区-南京市-江苏省-中国。

至此，实现了基于正向最长匹配和地址树推理的识别算法。

3)　基于规则的命名实体识别模块

a)　接口 String[] RegularParser.parse(Sting str)

输入：String str，地址树推理模块返回的剩余字符串(假定不存在区级及以上的地址)

输出：String ans[]. ans[1-10]保存识别结果。

b)　算法流程如下

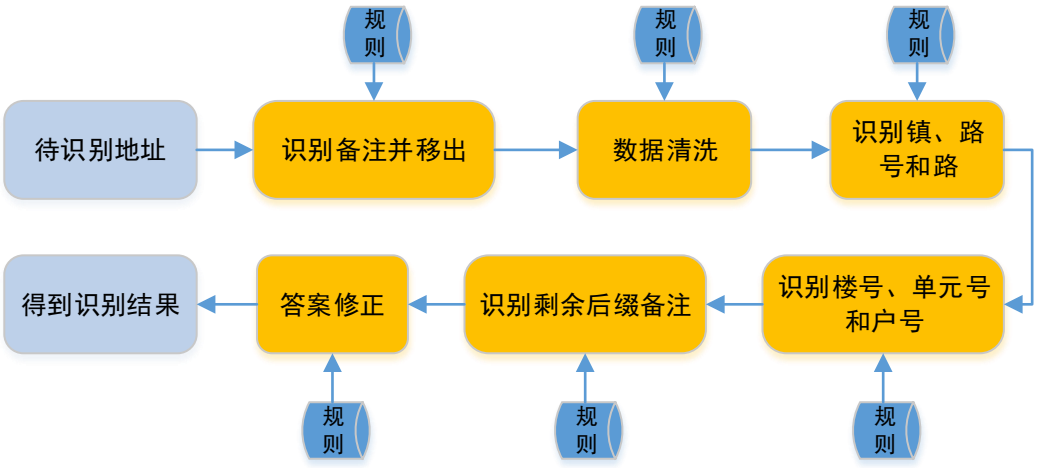


图6　基于规则的算法流程图

其中每一步的规则都经过了精巧的设计和充分的分析，具体内容在下一章详细介绍。

四、 关键设计

1) 地址树的推理算法

地址树推理算法依赖于正向最长匹配分词。具体流程已经介绍，不再阐述。

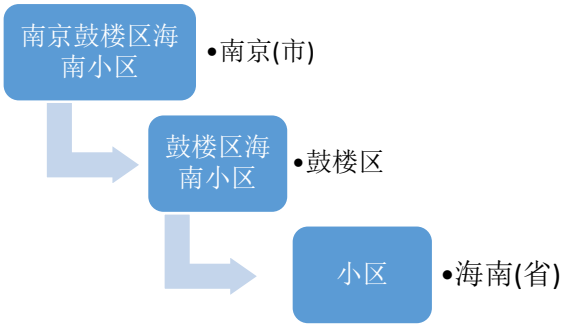
本节介绍具体实现时的一些参数设定和实现技巧。

a) 程序规定匹配的前缀长度至少是 **2**。

b) 程序要求识别出的地址实体的等级是严格递增的。

c) 在地址树上增加一个时间戳，使得每次利用地址树做分析，不需要重新初始化整棵树。

- 考虑“南京鼓楼区海南小区”这个例子。若不考虑等级递增，将会得到如下结果。

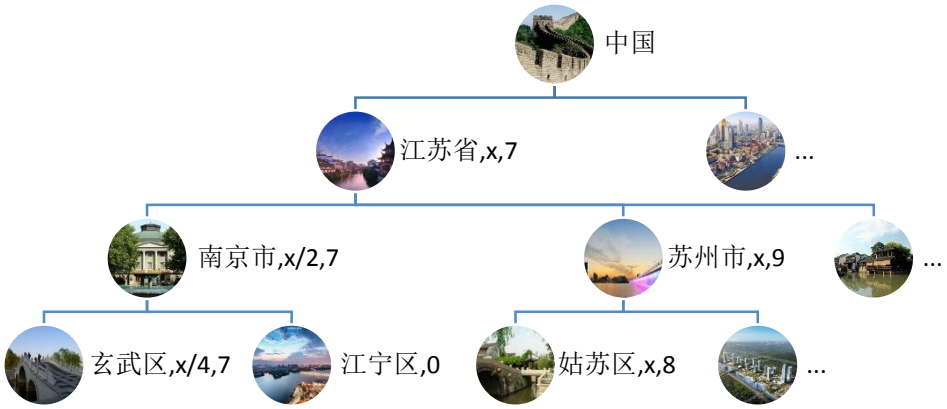


显然，不考虑等级递增，将会出现严重错误。

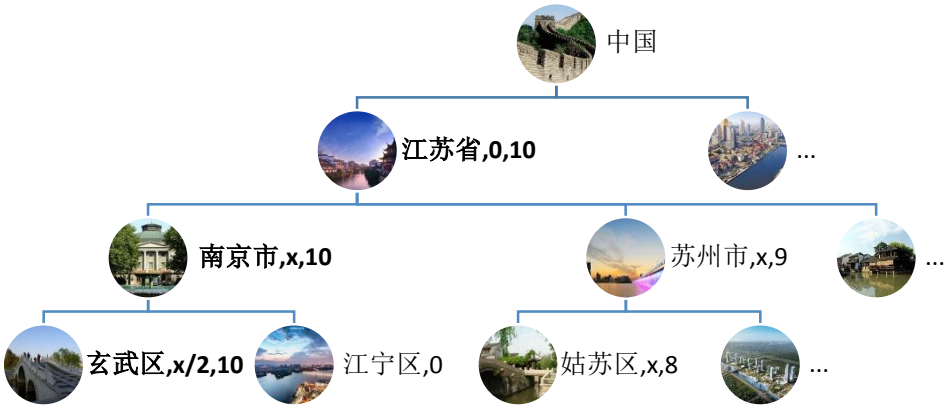
考虑等级递增后，可以解决这个问题。

- 在地址树上加上时间戳，提高多次查询速度。

依旧考虑“南京玄武区”的例子，第二维数字是时间戳。当前时间戳是 10，且还未使用正向最大匹配算法标记。



一旦访问树上的节点，先比较时间戳。若树上的时间戳与当前时间戳不一致，则先清空树上的数据，置时间戳为当前值。最终查询完成后，地址树状态变为：



通过此方法，正向最大匹配及地址树推理模型的时间复杂度为 O(|S|)，|S|表示源地址的长度。

地址树推理的得分如下图

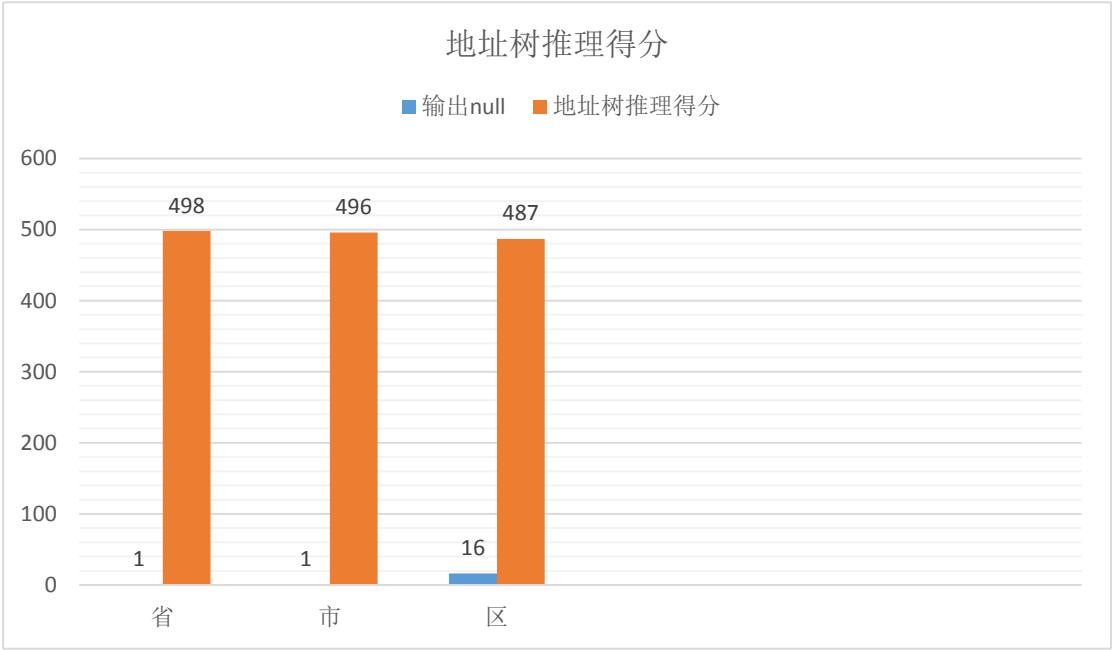


图7　地址树推理得分

2)　识别备注

有效备注识别包括 2 个规则。

第一是识别(*)中的备注，采用正则表达式(?<=\\()(.+?)(?=\\))

第二是在所有其他规则执行完毕后，剩余字符串判断是否为原串的后缀，若是且备注为空，则接受为备注。

识别出备注后将其移除。

2 种规则的得分分析如下图：

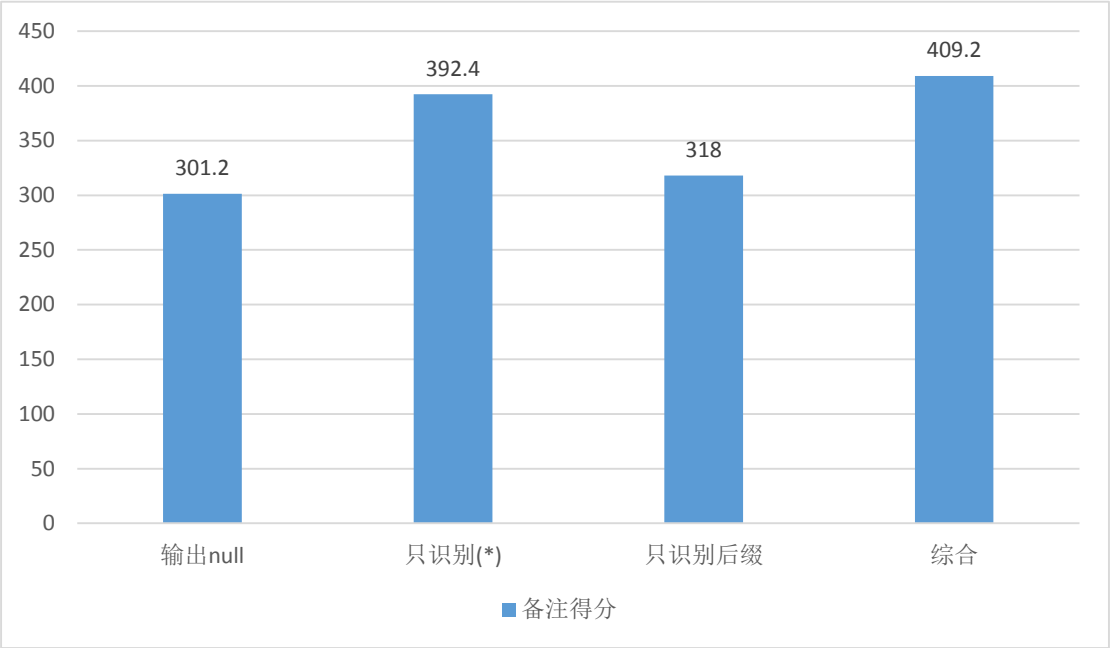


图8　备注识别得分分析

3)　数据清洗

在此简单将 某地区和休闲区 作为前缀的情况视为干扰去除。

正则表达式为`.+(?:地区|休闲区)`

4)　各级命名实体的识别规则

以下命名实体被识别后即被移除。

a)　镇和路的识别

识别镇的正则表达式为：

```
[\u4e00-\u9fa5_0-9]+?(?:路中段|街道办|街道|胡同|弄堂|街|路(?!口)|镇|开发区|子沟)
```

识别路的正则表达式为：

```
[\u4e00-\u9fa5_0-9]+{2,10}?(?:路中段|道中段|街道|胡同|弄堂|街|巷|路(?!口)|道|镇|大道|开发区|里)
```

b)　路号的识别

识别路号的正则表达式为：

```
(?:([付_附]{0,1}[0-9_-]+?(?:号院|号|弄))|([\u4e00-\u9fa5_0-9]+?(?:村|小区|社区|大厦|公寓|机场|广场|步行街|小镇|国际|公馆|购物中心|大楼|基地|雅居)))
```

c)　楼号、单元号、户号的识别

识别楼号的正则表达式为：

```
(?:([0-9_-_零_一_二_三_四_五_六_七_八_九_十]+?(?:栋|号楼|座|号院))|([\u4e00-\u9fa5_0-9]+{2,}?(?:商务楼|码头|村|小区|社区|大厦|公寓|机场|广场|步行街|小镇|国际|公馆|阁)))
```

识别单元号的正则表达式为：

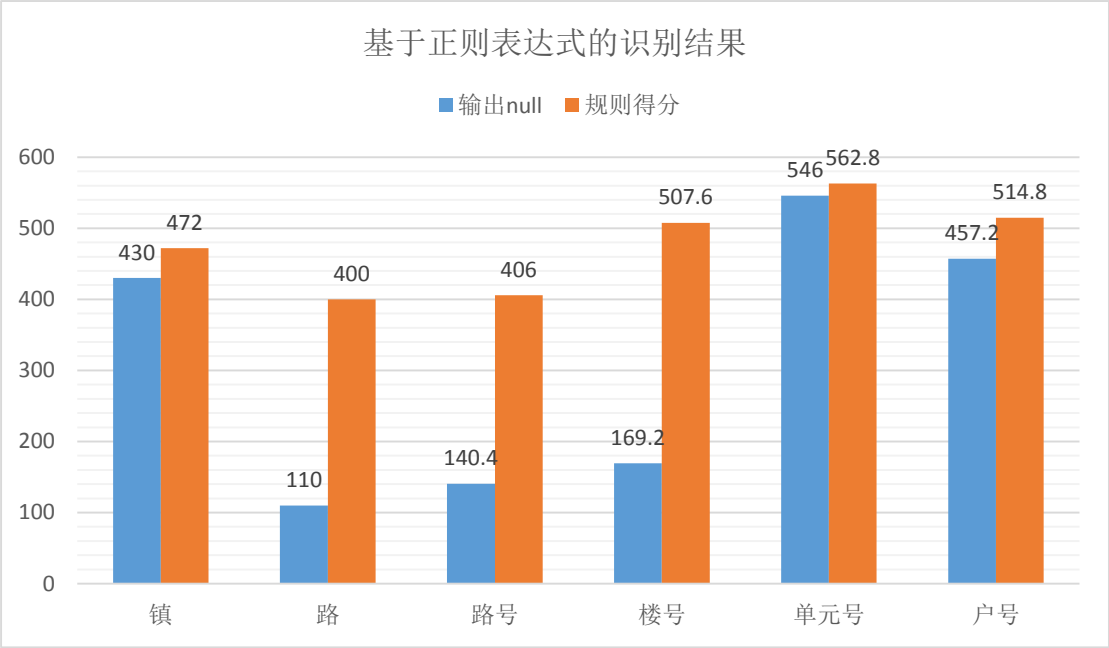
```
[0-9_-_零_一_二_三_四_五_六_七_八_九_十]+?(?:单元|栋|幢|号楼|座|号院)
```

识别户号的正则表达式为：

[0-9_-_零_一_二_三_四_五_六_七_八_九_十_A-Z_a-z_._-]+?(?:单元|号楼|号铺|座|室|号|楼|#|门面|店面)

d) 答案清洗

由于某些答案出现了重复前缀的情况，例如“西塘西塘镇”，故将其修正。



5) 小结

基于地址树推理的模型得到了很好的结果。基于规则的识别方法对某些级别的地址识别存在一定问题。

由于算法系统的复杂性，上述介绍略去了一些细节。整个算法系统的具体实现请参考源代码。

五、 实验分析

1) 运行平台

表1 实验运行环境

类型	软硬件
操作系统	Windows 7 Ultimate
CPU	Intel Core i5-4590
内存	16G
开发工具	Eclipse 4.5.1
Java 版本	Java 1.8.0

类型	软硬件
依赖库	Commons CSV 1.2

所有代码均在以上平台环境运行。

2) 算法性能

表2 算法系统精确度

数据集	训练集	A 榜	B 榜	1 级	2 级	3 级	4 级
分数	4753.4	4770.8	4755.6	498	496	487	472
数据集	5 级	6 级	7 级	8 级	9 级	10 级	
分数	400	406	507.6	562.8	514.8	409.2	

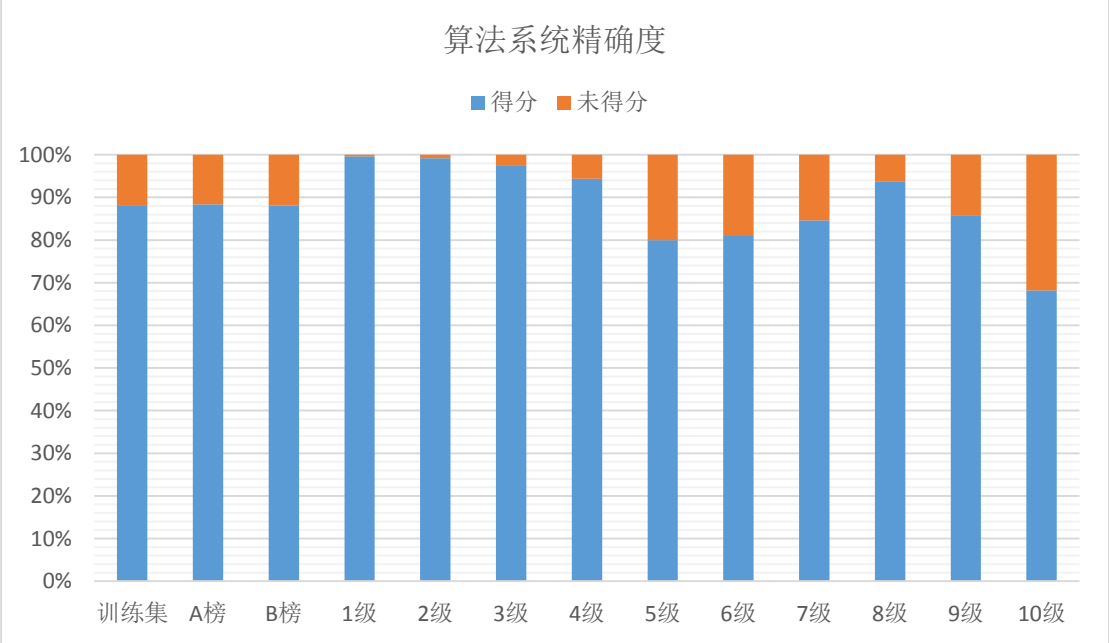
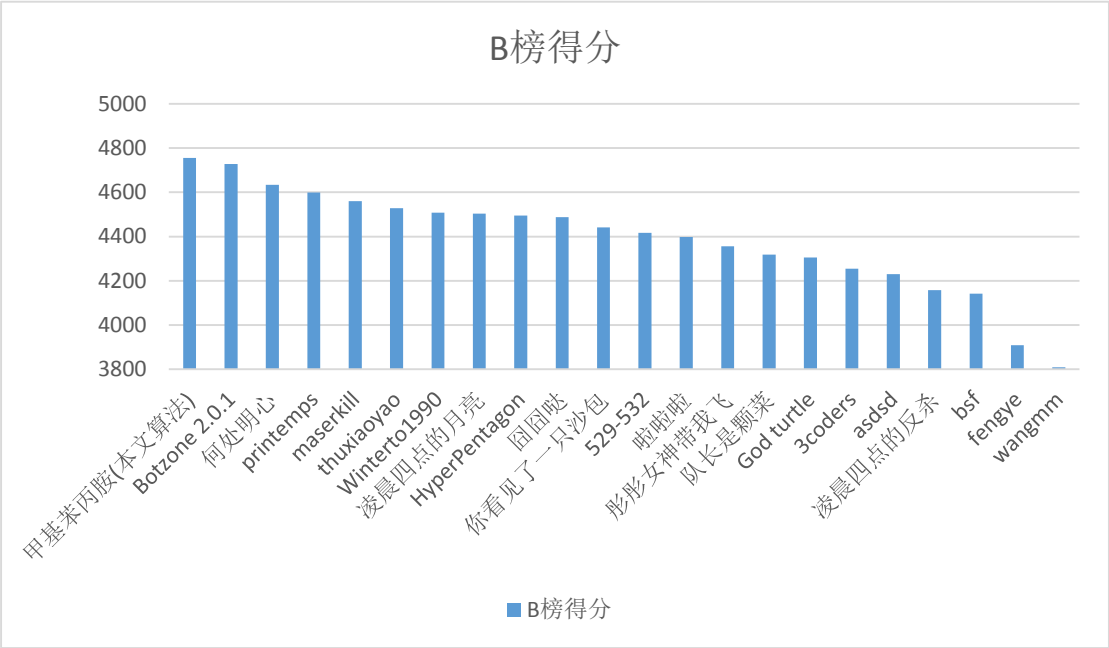


图9 算法系统精确度

与大赛上其他队伍比较，本算法系统的成绩是第一名。



分析：

本系统使用的算法，在大赛的 B 榜成绩取得了第 1 名的成绩。

系统对前 3 级地名的识别率已经接近 100%。其中省级的 2 个识别结果与训练集不同，是由于训练集的标注错误。可以认为省级结果全部正确。

除去训练集的错误标注，市级只有“重庆市辖区”未推理出来。该模式的推理方式可以归纳为：在地址树上标记了重庆市，且重庆市只有 1 个子节点，因此可以推理为重庆市辖区。经过分析，此推理模式合理，有待继续完善。

区级(3 级)的识别精度略有下降。认为造成下降的原因主要是训练集对某些地区的记法与最新版国家统计局公布的数据不一致。同时训练集的推理方式考虑到镇级别的信息；考虑到算法系统的前提假定及稳定性，本系统只利用前 3 级信息进行推理。

乡镇级(4 级)的识别精度为 94.4%。由于 4、5 级的数据来自网络，性能的降低可能来自于算法使用数据和训练集使用数据的不一致性。单纯使用地名数据做查询，分数为 455。利用了正则匹配的方法识别未登录地名之后，分数提高至 472。

5、6 级地址识别任务是识别出路和路号。算法的正确率下降为 80%。正确率明显下降的原因可能是路和路号的复杂性。例如大部分的路都以“路”“街”为后缀，但更复杂的模式难以通过简单的模式匹配方法分析。改进方法可以是分析出覆盖率更全的地址模式，也可以借助机器学习的模型自动分析。

7~9 级的识别任务要求识别出楼号、单元号和户号。虽然这几项任务看似简单，但对单元号和户号单独分析，可能会存在歧义。例如 10 楼可能是单元号，也可能是户号；但 335 楼是户号的可能就不大（太高）。对于此类复杂的情况，算法系统只能尽力考虑周全。此模块最终的识别精度为 88.1%。

备注(10 级)的识别精度依赖之前的模块。利用(*)和剩余后缀规则，正确识别出的备注有341 条，这对于复杂的备注标注任务，已经是不错的成绩了。

表3　算法效率

中文地址记录	运行时间
50 万条	18.5 s
平均每条	37us

3)　实验小结

通过实验比较，认为本文实现的算法系统在精确度和效率上具有明显的优势。在中文地址大赛 B 榜成绩排行中排名第 1。分析每一条记录仅需 37us。

六、 不足与改进

本文实现了结合地址树推理和正则匹配的中文地址识别算法系统。但由于经验和时间的不足，未能使用机器学习算法进一步提高系统的正确率。因此，还有如下几个方面可以改进：

1) 结合概率模型进行分析。

目前中文分词常用的有条件随机场和隐式马尔可夫模型。在获得大量的训练数据的基础上，可以用这些机器学习的方法进一步提高算法正确率。

2) 利用无监督的方法自动发现中文地址的结构化信息。

从数据上看，无标注的数据相比于有标注的数据多很多。因此，如果能深度挖掘无标记的数据，就能创造更多数据价值。

七、 参考文献

1. 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[J]. 电子学报, 2015(08):1481-1487.
2. 俞鸿魁, 张华平, 刘群,等. 基于层叠隐马尔可夫模型的中文命名实体识别[J]. 通信学报, 2006, 27(02):87-94.
3. 向晓雯, 史晓东, 曾华琳. 一个统计与规则相结合的中文命名实体识别系统[J]. 计算机应用, 2005, 25(10):2404-2406.
4. 向晓雯. 基于条件随机场的中文命名实体识别[D]. 厦门大学, 2006.
5. 赵琳瑛. 基于隐马尔科夫模型的中文命名实体识别研究[D]. 西安电子科技大学, 2008.
6. 科曼. 算法导论[M]. 机械工业出版社, 2006.