 DOI: https://doi.org/10.12345/smg.v5i3.12769

Research on Object-oriented Geocoding and Address Matching System

Jie Tang

Xinjiang Highway Plannig Survey Design and Research Institute, Urmqi, Xinjiang, 830006, China

Abstract

The paper is based on object-oriented programming methods, distinguishing between geocoding as a database object and address matching as a method object. The system design is completed through the definition of subclasses, building a bridge between spatial and non spatial data. Unlike process oriented programming, this system has stronger maintainability and scalability, higher computational efficiency, and better program scale.

Keywords

geocoding; address matching; object-oriented; process oriented

面向对象的地理编码与地址匹配系统研究

唐洁

新疆维吾尔自治区交通规划勘察设计研究院，中国 · 新疆 乌鲁木齐 830006

摘　要

论文基于面向对象的编程方法，区分地理编码为数据库对象，地址匹配为方法对象，通过子类的定义完成系统的设计，搭建了空间数据和非空间数据进行联系的桥梁。区别于面向过程的编程方式，该系统具备更强的易维护性和可扩展性，更高的运算效率以及更优的程序规模。

关键词

地理编码；地址匹配；面向对象；面向过程

1 引言

城市在规划、建设和管理过程中会产生大量的业务数据，虽然其中 80% 以上的数据与地理位置相关，但是这些信息并没有关联坐标，成为非空间数据，因此无法通过整合进行空间分析。随着地理信息系统（GIS）在中国数字城市建设中的应用日趋广泛，空间数据和非空间数据的整合与共享愈发重要，成为城市基础设施建设的不可缺少的重要内容之一。将空间数据和非空间数据联系起来的重要桥梁就是地址匹配，它是基于空间定位技术的一种将描述性的地址位置信息转换为 GIS 系统可以认知的地理坐标的方法。在传统的理解范畴中地址匹配又称地理编码，主要研究方向集中在如何让 GIS 系统准确认知一个地址并与相应的地理坐标进行匹配。但是论文基于面向对象的编程理论，首先将地理编码和地址匹配区分为两个概念 [1]。

2 面向对象

面向对象（Object-oriented）是建立在“对象”概念基础上应用于软件开发过程中的系统方法。对象是由数据和容许的操作组成的封装体，与客观实体有直接对应关系，一个对象类定义了具有相似性质的一组对象。所谓面向对象就是基于对象概念，以对象为中心，以类和继承为构造机制，来认识、理解、刻画客观世界和设计、构建相应的软件系统。面向对象的核心是“开发对象模型”，包括四大主要要素即抽象、封装、模块化、层次结构和三大次要要素即类型、持久、并发。面向对象就主要围绕这几个要素进行的，最难的部分是确定正确的类和对象。

与面向对象相对应的是面向过程（Process-oriented）[2]，是一种以过程为中心的编程思想。本文研究的系统，在传统的理解范畴中地址匹配等同于地理编码，系统实现如图 1 所示。

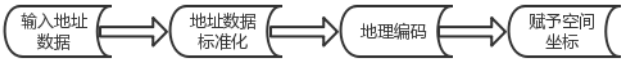

图 1 系统实现过程

根据图 1，首先分析出解决问题所需要的步骤（即“第一步做什么，第二步做什么，第三步做什么”），然后用函数实现各个步骤，再依次调用，是一个典型的面向过程的思

【作者简介】唐洁（1986–），女，中国陕西宝鸡人，硕士，从事项目管理和地理信息系统等方面的研究。

维。在实现步骤中对于地址数据标准化处理成为一个核心问题，其中分词技术的研究成为重点。但是基于汉语相较于英语的特殊性，并不存在某一种单一的分词方法可以做到准确无误，即便加入其他的方法予以校正，仍无法做到完全准确。在系统设计时，增加了程序的重复和冗余，进而影响系统响应。并且在系统出现任何系统 Bug 时，都需要对整个过程进行梳理和纠错，降低了易用性。

因此将论文研究系统从思路上拆解为地理编码和地址匹配两个大的对象，通过类的定义，以面向对象的编程思维进行程序设计能有效避免因分词技术的不足所导致的系统问题 [3]。

3 地理编码

首先将地理编码定义为数据库对象，其次将地理位置拆分成不同的子对象也就是编程时的子类，最终实现统一编码。完整的地理编码拆分如下 7 个子类。

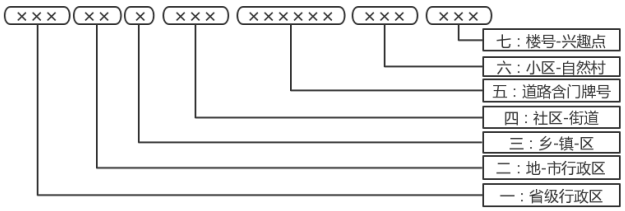

图 2 地理编码的拆分

如图 2 所示将地理编码的过程根据行政区划的高低顺序拆分为 7 个子类对象，其中省级行政区、地—市行政区和乡—镇—区 3 个子类执行 GB/T 2260 规定的六位数字代码，社区—街道的编码按照所在地的要求执行相关地方标准。属性中的空间坐标为区域几何中心的坐标。这 4 个子类的属性定义与字段含义见表 1。

表 1 属性定义表（一）

	省级行政区	地—市行政区	乡—镇—区名	社区—街道
定义	Class Province	Class Pro_city	Class District	Class Community
编码	Province.code	Pro_city.code	District.code	Community.code
地名	Province.name	Pro_city.name	District.name	Community.name
经度	Province.long	Pro_city.long	District.long	Community.long
纬度	Province.lati	Pro_city.lati	District.lati	Community.lati
上级地址		Pro_city.Superior	District.Superior	Community.Superior

如图 2 所示，第五子类道路含门牌号的编码在本行政区域内按照起点由东向西和由南向北的顺序编号，参照 GB/T 21381—2008 标准执行。属性中的空间坐标为道路中心线的坐标。城市道路门牌号的编码一般有如下两种情况：

①道路两侧均有门牌号，按照自东向西、自北向南，左双右单的顺序连续递增排序。

②道路只有一侧有门牌号，门牌号递增排序。

因此属性定义需要增加一个类型指针，以判断语句指向不同的对象属性。本子类需要定义两类属性，具体属性定义与字段含义见表 2 和表 3。

表 2 属性定义表（二）

	道路含门牌号
定义	Class Road
编码	Road.code
道路名	Road.name
上级地址	Road.Superior

表 3 特殊属性定义表

类型	Road.type	
	Road.type=single	Road.type=double
起点门牌号	Road.from_num	Road.left_from_num
		Road.right_from_num
终点门牌号	Road.to_num	Road.left_to_num
		Road.right_to_num
起点位置	Road.from_long	Road.left_from_long
		Road.right_from_long
	Road.from_lati	Road.left_from_lati
		Road.right_from_lati
终点位置	Road.to_long	Road.left_to_long
		Road.right_to_long
	Road.to_lati	Road.left_to_lati
		Road.right_to_lati

如图 2 所示，第六子类小区—自然村编码在本行政区域内按照由东向西和由南向北的顺序编号；第七子类楼号—兴趣点编码在小区、自然村或街巷内按照起点由东向西和由南向北的顺序编号，编码位数各地可根据实际情况进行扩充。如两位编号为 01~99，三位编号为 001~999，以此类推。属性中的空间坐标为区域几何中心的坐标。这 2 个子类的属性定义与字段含义见表 4。

表 4 属性定义表（三）

	小区 - 自然村	楼号 - 兴趣点
定义	Class Cell	Class POI
编码	Cell.code	POI.code
地名	Cell.name	POI.name
经度	Cell.long	POI.long
纬度	Cell.lati	POI.lati
门牌号	Cell.number	POI.number
上级地址	Cell.Superior	POI.Superior

根据以上 7 个子类的属性定义进行数据库对象的开发，通过子类的继承和方法调用进行相互的关联，构成完整的地

理编码对象系统[4]。

4 地址匹配

首先将地址匹配定义成方法对象，目的是将一条地址信息从地理编码数据库对象中调取处理，完成标准化、录入、查询和分析等功能。其次将该方法对象分为高级别匹配和低级别匹配 2 个子类。

高级别匹配子类是针对行政区划级别较高的地址，对应数据库对象中省级行政区、地—市行政区、乡—镇—区、社区—街道和道路含门牌号 5 个子类。该子类中主要封装数据库对象调用和地址选择 2 个函数。

数据库对象调用函数实现对于地理编码中数据库对象的调用并列表显示的功能，地址选择函数实现用户在列表显示中直接选择对应地址的功能。通过在主程序中设置 5 个实例，实现图 2 中第一至第五级别的部分地址匹配并标准化。

低级别匹配子类是针对具体到点的地址，对应数据库对象中和楼号—兴趣点 2 个子类。当地址匹配到达图 2 中第六和第七子类时，表达趋于复杂性，同时数据库规模也呈指数增长，因此不能通过调用数据库子类让用户直接进行选择的方式实现。在程序设计上复杂性高于高级别匹配子类。

该子类主要封装了 4 个函数，其中数据库对象调用函数实现对于地理编码中数据库对象的调用；地址输入函数实现用户输入内容的接收。地址检查函数和坐标计算函数相对复杂。

①地址检查函数是针对用户输入字段与数据库内标准字段存在偏差设计的，主要分为两种情况，例如：第一种情况，标准字段为“迎宾路”，用户输入“迎宾”，定义为字段缺省；第二种情况，标准字段包括“北京南路”和“北京东路”等，用户输入北京路，定义为后缀缺省。

函数主要通过“while 循环”语句将用户输入与数据库记录相近的字段进行逐字比对，用“if 判断”语句检查匹配度，若大于 60%，则认为找到标准字段，针对字段缺省的情况，将找到的标准字段显示并由用户最终确认；针对后缀缺省的情况，将找到的标准字段列表显示并有用户进行选择确认。通过该函数对用户输入的非标准字段完成地址标准化和地址匹配。

②坐标计算函数是针对在空间数据库不完备导致空间坐标无法匹配的情况下，通过数据库中临近地址的空间坐标进行坐标推算。计算方法采用平均值法，以等分的方式对临近地址进行处理，将建筑物假设为质点，街道假设为直线。设用户输入的待匹配点为 $P_0(X_0, Y_0)$，前一临近地址为 $P_1(X_1, Y_1)$，后一临近地址为 $P_2(X_2, Y_2)$，其中 P_0、P_1、P_2 为门牌号，则有公式（1）：

$$\begin{aligned} X_0 &= X_1 + (X_2 - X_1)(P_0 - P_1)/(P_2 - P_1) \\ Y_0 &= Y_1 + (Y_2 - Y_1)(P_0 - P_1)/(P_2 - P_1) \end{aligned} \quad (1)$$

如需匹配北京南路 320 号的坐标，则可查询到的临近地址为北京南路 318 号和北京南路 322 号，代入公式（1），即可求出待匹配点的空间坐标。

在主程序中设置 2 个实例，实现图 2 中第六至第七级别的部分地址匹配并标准化，最终可以完成对于用户输入地址的匹配，搭建起空间数据与非空间数据之间的桥梁，通过地理信息系统完成显示、录入、查询、统计及分析等功能[5]。

5 总结

论文设计的系统因采用面向对象的编程，以对立子类封装代码，当数据库对象或方法对象改变时，只需重写部分需要变化的子类代码，无需改动整个系统代码，方便维护，提升扩展性。

由地址匹配的方法对象可以看出，面向对象的编程可以进行归类总结，仅定义 2 个子类便可完成 7 级地址的实例化，程序规模较小。而地理编码数据库对象，虽部分属性定义内容相似，但为了方便后期维护并没有进行子类的归类处理。

论文涉及的子类、属性、方法和代码编写方式均源自 python 语言，使用其他面向对象的语言时，思路相通，但仍需注意代码细节。

论文设计系统仍有不足，比如人工干预较多，后期需要利用 python 语言在人工智能方面的优势，在系统自动化方面进一步完善。

参考文献

[1] Duck-HyeYang, Lucy Mackey Bilaver, Oscar Hayes, et al. ImprovingGeocoding Practices: Evaluation of Geocoding Tools[J]. Journal of Medical Systems, 2004(28):4.

[2] 朱建伟,王泽民.地址编码原理及其本地化解决方案[J].北京测绘,2004(2):24-27.

[3] 朱纯阳.Python操作SQLite数据库[J].电脑编程技巧与维护,2015(15):65-66.

[4] GB/T 2260 中华人民共和国行政区划代码[S].

[5] GB/T 21381—2008 交通管理地理信息实体标识编码规则 城市道路[S].