地理空间地址解析技术

**关键技术研究报告**

撰写人 ：王秦豫

二零一七年九月

# 1 简介

## 1.1 背景

自然语言处理（NLP）是计算科学领域与人工智能领域中的一个重要方向，它研究人与计算机之间通过自然语言进行有效通信的理论和方法。随着各行业信息化程度的提升，自然语言的识别与处理在信息化建设中占据越来越重要的地位。其中地理空间地址的解析技术就属于自然语言处理相关范畴。地理空间地址在不同的行业领域有着不同的展现方式，人们往往需要人工的去识别不同类型的输入地址。地址的多变性和多样性导致了计算机在代替人工操作时产生各种错误和误差，有时结果甚至无法被使用，导致数据浪费。因此更高效稳定的解析技术和算法尤为重要。比如在物流行业中地址的存储和识别技术可以使得从各种渠道收集的货物地址得到规整，并应用于之后的分发工作。这解决了每天亿级的物流人力成本。在医疗行业，医疗信息也包含大量地址，地址的识别、清洗、匹配以及应用也能加快信息化的程度，减少人力成本。本文主要以医疗信息行业为例，对地理空间地址的解析技术及其应用。

## 1.2 概要

本文所研究的地理空间地址解析技术（以下简称“地址解析”）主要是为了提高医疗基础信息的准确度和可应用度，该技术可服务于各类医疗信息平台（或系统），为平台（或系统）中地址信息相关部分提供结构化解决方案。在不同的医疗机构、医疗工作人员以及居民（含“患者”）等输入端输入到系统中的地址不尽相同，数据间差异性和误差较大，存在大量非结构化的地址、缺失地址和错误地址。为了将数据进行统一化存储，需要利用地址解析将以上数据修正为标准的结构化地址数据。

地理空间地址解析技术主要分为中文分词、数据清洗、词条匹配等模块对数据依次进行操作，在操作前需要在系统中生成基本地址词库，地址词库的样式和结构不得随意更改和删除，地址词库的丰富度与词条匹配模块的准确性正相关，与结果的有效性息息相关。本文对技术进行了Java语言实现，采用的地址库为中国国家统计局行政区划代码统计标准（版本：2016年7月）。中文分词过程采用了Aho Corasick (AC)自动机结合DoubleArrayTrie（双数组Trie树）极速多模式匹配算法，数据清洗和数据匹配采用了HashMap循环检索算法。经处理可得到符合要求的结构化地址。算法具有缺级修复、父子验证、错误反馈、高并发处理、一次加载多次使用的优点。其准确率和性能也优于我国现阶段医疗行业普通算法。

## 1.3 结构

地理空间地址解析技术包含词库加载、中文分词、数据清洗、词条匹配、结果装载五部分，各部分之间采用正向顺序进行。在程序启动后，词库只需加载一次即可。词库使用Mysql数据库进行存储。地址解析结果可被其它程序直接使用。具体流程如图1所示。

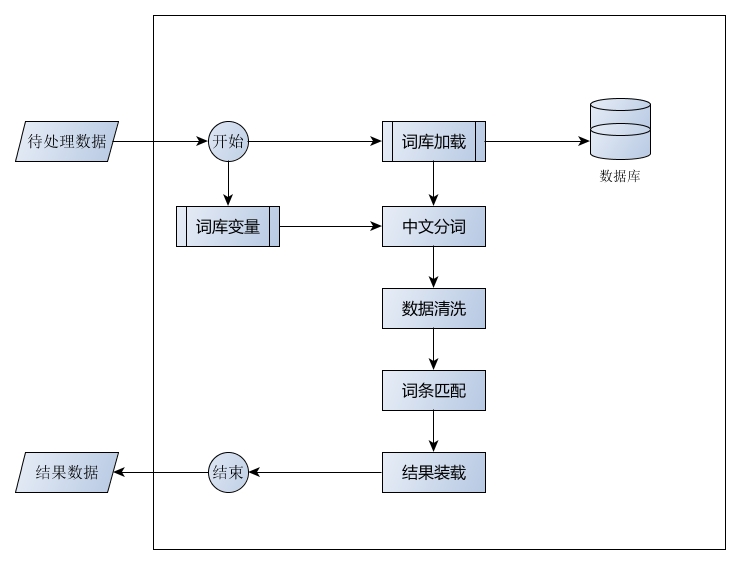


图1：地址解析技术流程图

# 2 突破的关键技术或技术要点

## 2.1 中文分词技术的实现

中文分词技术采用Aho Corasick (AC)自动机结合DoubleArrayTrie（双数组Trie树）极速多模式匹配算法实现。其基本原理是为一个双数组Trie的每个状态（具体体现在下标）附上额外的信息。因为AC自动机的基础（success表）就是Trie树，所以额外的增加output表和fail表即可。基于此理论，就是为每个状态（和）构建和。

### 2.1.1双数组Trie树简介

双数组Trie树是一种空间复杂度低的Trie树，它是Trie结构的压缩形式，仅用两个线性数组来表示Trie树，该结构有效地结合了数字搜索树检索时间高效的特点和链式表示的Trie空间结构紧凑的特点，双数组Trie树实际上是一个确定有限状态自动机（DFA），每个节点代表自动机的一个状态，根据变量不同进行状态转移。当到达结束状态或无法转移时，完成一次查询操作。在双数组所有键中包含的字符之间的联系都是通过简单的数学加法运算表示，不仅提高了检索速度，而且省去了链式结构中使用的大量指针，节省了存储空间。单一使用双数组Trie树能高速的完成单串匹配，并且内存消耗可控，但如果匹配多个模式串则需要先实现前缀查询，然后频繁的截取文本后缀才可多匹配，这样一份文本要回退扫描多遍，性能很低。如果使用AC自动机来搭配使用，就能结合两者的优点，提高性能。为此我们需要先行构建双数组Trie树，然后再构建AC自动机。

### 2.1.2 双数组Trie树构建

双数组Trie树的构建是先构造一个Trie树，然后把它压缩到两个数组上面。两个数组分别为base和check，它们的值都代表一个确定的状态，base存储当前的状态以供状态转移使用，check验证字符串是否由同一个状态转移而来并且当check为负时代表字符串的结束。假定有一个字符串的状态s，当前的字符串状态为t，假定t加了一个字符串c就等于状态tc，加一个字符串x就等于状态tx，那么有，







在每个节点插入的过程中会修改这两个数组。具体步骤如下：

1. 初始化root节点，。
2. 对于每一群兄弟节点，寻找一个begin值使得，也就是找到了n个空闲空间是siblings中的n个节点对应的code值。

int pos = siblings.get(0).code;

**While** (**true**)

{

pos++;

begin= pos - siblings.get(0).code; *// 当前位置离第一个兄弟节点的距离*

}

1. 然后将这群兄弟节点的check设为，则叶子节点i的的值一定等于i，因为它是兄弟节点中的第一个，并且它的code为0。

check[begin + siblings.get(i).code] = begin;

1. 对每个兄弟节点，如果它没有子节点，令其base为负值；否则base等于该节点的子节点插入位置（begin值），同时插入子节点。迭代跳转到步骤（2）。

**if** (fetch(siblings.get(i), new\_siblings) == 0) *// 无子节点，也就是叶子节点，代表一个词的终止且不为其他词的前缀*

{

base[begin + siblings.get(i).code] = -siblings.get(i).left - 1;

*//……*

}

**else**

{

int h = insert(new\_siblings); *// dfs*

base[begin + siblings.get(i).code] = h;

}

### 与单独构建双数组Trie树不同，需要在为一个Trie树state创建的时候，让该state记住自己的i，这样就建立了state和下标的映射。

### 2.1.3 AC自动机的构建

自动机按照文本字符顺序，接受字符并发生状态转移。这些状态缓存了“按照字符转移成功（但不是模式串的结尾）”，“按照字符转移成功（是模式串的结尾）”，“按照字符转移失败”三种情况下的跳转与输出情况，因而降低了复杂度。AC自动机的三个核心函数分别为success、fail和emits。

Success函数表示成功转移到另一个状态（goto表或者success表），fail函数表示不可顺着字符串跳转的话，则跳转到一个特定的节点（fail表），从根节点到这个特定的节点的路径恰好是失败前的文本的一部分，emits函数表示命中一个模式串（output表）。其中success表由双数组Trie树构建完毕，现在只要在添加fail表和output表即可。

在构建AC自动机时，每构建一个节点state的fail表，就利用state映射的下标state.id将设为failState.id，对于output表也同理。

## 2.2 匹配技术的实现

# 3 研究成果及创新点

# 4 展望

# 5 引用