**lucene (42)源代码学习之FST(Finite State Transducer)在SynonymFilter中的实现思想在Lucene4中，很重要的一种数据结构就是FST(Finite State Transducer),又称Mealy Machine。**

**在SynonymFilter的实现中，FST可以用HashMap代替。但是相比HashMap，FST有以下优点：**

**1、紧凑的结构，通过对词典中单词前缀和后缀的重复利用，压缩了存储空间。**

**2、O(len(str))的查询时间复杂度。**

**如果不考虑FST的输出，FST本质上是一个最小的，有向无环DFA。**

**经过差不多一个星期的学习，终于对Lucene中FST的构造过程有了初步的了解。为了简化问题，我们把FST的输出都置为相同的因子。**

**案例：**

**对aaaa,bbaa,ccbaa,ddcbaa这4个单词构建FST。(注意：单词插入前必须先进行排序，否则就无法生成最小FST)**

**先给出最终的结果：**

**(注：这副图来自于http://examples.mikemccandless.com/fst.py)**

**那么其构造的过程是怎么样的呢？**

**我们从Lucene42的org.apache.lucene.util.fst.Builder类的add()方法作为切入口。**

**对于Lucene42的源代码，有这样几点需要理解。**

**1、UnCompiledNode类和CompiledNode类。**

**UnCompiledNode可以理解为没有编入FST的结点，CompliedNode为已经编入FST的结点。在Builder类中，初始化了一个UnCompiledNode数组。这个数组中存储的是什么呢？论文《Direct Buildingof Mimimal Automation for A Given List》中描述构造一个伪最小FST的方法：**

**假设有{w1,w2....,wn} n个单词。**

**a、先构造一个除单词w1外，最小的FST。（此时FST中有w1一个单词）**

**b、构造一个除w2外，最小的FST。（此时FST中有w1,w2两个单词）**

**c、构造一个除w3外，最小的FST。（此时FST中有w1,w2,w3三个单词）**

**d、构造一个除w4外，最小的FST。（此时FST中有w1,w2,w3,w4四个单词）**

**e、构造一个除ε(空串)外，最小的FST。此时FST最小。**

**这其实就是一种迭代的思想。UnCompiledNode存储的就是上文提到的单词w1，w2……。**

**2、freezeTail方法的作用是什么？**

**freezeTail方法的作用就是把UnCompiledNode进行Compile成CompiledNode。也就是构造一个除wi外，最小的FST。通过方法名称，也可以理解为把后缀(tail)固定(freeze)下来。**

**看论文不如走代码，我们一步步看代码是怎么走吧。**

**第一步：往FST中插入第一个单词aaaa。**

**插入的代码如下：**

**此时FST的结构如下：**

**(状态的编号代表Builder中frontier数组的下标；红色的箭头代表是结束边)**

**上图的FST即是除字符串aaaa外最小的FST，（其实最小FST为空）。**

**第二步：往FST中插入第二个单词bbaa。**

**在插入之前，需要对aaaa进行freezeTail，即把除aaaa和bbaa前缀之外的部分compile到最小FST中。**

**Compile的过程是从后往前的，也就是先处理State4，然后处理State3……**

**FST把单词的后缀存储到HashNode这种数据结构中去，如果新插入的字符串在compile过程中，与前面的单词有公共后缀，则直接使用其后缀，而不创建新的结点。**

**其过程依次如下：(具体的代码参考freezeTail函数)**

**freezeTail的过程中，把frontier的位置空出来了。然后将新的字符串插入到frontier中。**

**第三步：往FST中插入ccbaa。**

**在插入ccbaa之前，仍然需要将bbaa进行freezeTail。**

**由于aaaa和bbaa有{a},{aa}这两个共同的后缀，所以C2 、C4节点可以重复利用。**

**这个过程就是在利用后缀对状态进行压缩。**

**第四步：往FST中插入ddcbaa.**

**在插入之前对ccbaa进行freezeTail操作。**

**有了前面的基础，这里就只画出来了它的变化。经过freezeTail后，frontier数组又空出来了。接下来就把ddcbaa插入到FST中去。**

**最后一步，Builder.finish()。**

**在这一步中，会调用freezeTail(0)。这样的话，整个FST就变成了一个最小的，无环FST了。**

**将两副图进行对比：**

**是不是一样的？**

**图中状态里面的数字，如 -1, 2，4 是程序处理过程中后缀存储在NodeHash中的位置。可以这样理解，FST用NodeHash存储了字符串所有可能的后缀状态。**

**这里的分析过程，忽略了FST对output的处理。Output对FST的结构是没有影响的。**

**FST作为一种数据结构，在自然语言处理中应用非常广泛，比如语音识别，机器翻译等。在Lucene中也有广泛的应用，比如同义词处理、FuzzyQuery、拼写检查、自动补全等。其理论基础就是有限状态机。本文略过了论文中数学证明等知识点，(其实是我自己也没弄清楚 )从Lucene源代码实现的过程对FST的构造思想进行追踪。**

**参考资料：**

**1、 Build your own finite state transducer**

**(http://blog.mikemccandless.com/2013/06/build-your-own-finite-state-transducer.html)**

**2、Lucene&solr4 实践（4）(http://blog.sina.com.cn/s/blog\_4d58e3c001016gg4.html)**

**3、《自动机理论、语言和计算导论(第二版)》**

**4、《编译原理》**

**5、《Direct Building of Mimimal Automation for A Given List》**

**6、《Direct Construction of Mimimal Structure Acyclic SubsequentialTransducers》**