毕设工程说明文档

丘骏鹏

1 工程概况

毕设的工作共有两个工程，分别为 TemplateExtractor 和 TemplateExtractorDemo，实现语言都是 Scala，开发环境为 Eclipse。其中，Scala 的版本为 2.10.1，由于 Scala 本身不向后兼容，因此必须保证所有的代码都采用同一版本进行编译。所有的程序仅保证可以在 Linux 下运行。

注：很多 Linux 发行版的仓库中的 Scala 主版本号还是 2.9（如我使用的 Ubuntu 13.04），因此我是手动安装的 Scala 2.10。

我在我的服务器账号 qiujunpeng 目录下手动安装了需要的软件，并配置好了环境变量，因此使用 qiujunpeng 账号登陆可以直接使用各个软件。

TemplateExtractor 为主要的工程，实现了模板的自动聚类和抽取；TemplateExtractorDemo 比较简单，实现了模板匹配的演示 Web Service。

2 TemplateExtractor 介绍

2.1 代码管理综述

代码的编译和管理采用 SBT(Simple Build Tool)，是类似 Java 的 Maven 的一个代码自动构建工具，详细的介绍可以看`http://www.scala-sbt.org/`。SBT 的主要配置文件为 build.sbt，置于工程根目录下，下面以 PROJECT_DIR 表示项目的根目录。所有工程中的依赖关系，即用到的所有第三方库，包括 twitter util，akka，jsoup，icu4j 等都统一使用 SBT 进行管理。使用的 SBT 版本为 0.12.0。

2.2 SBT 使用方法

以下说明 SBT 的使用方法：

- 编译、运行代码：修改根目录 PROJECT_DIR 下的 build.sbt 中的最后一行：

  ```
  mainClass in (Compile, run) := Some("path.to.main.class")
  ```

  将其中的 `path.to.main.class` 修改为需要运行的类的路径。在根目录 PROJECT_DIR 键入 `sbt` 命令，即可进行 SBT 的控制台。在 SBT 的控制台中，输入 `compile` 编译，然后输入 `one-jar` 导出为 Jar 包（需要安装 SBT 的 sbt-onejar 插件，工程地址为`https://github.com/retronym/sbt-onejar`）。进入 SBT 的控制台后，输入 `one-jar` 即可导出 Jar 包。导出的 Jar 包有两个：

  ```
  PROJECT_DIR/target/scala-2.10/templateextractor_2.10-0.1-SNAPSHOT.jar
  PROJECT_DIR/target/scala-2.10/templateextractor_2.10-0.1-SNAPSHOT-one-jar.jar
  ```

  第一个包含工程所有的源代码编译成的.class 文件，第二个则包含了第一个 Jar 包和所有的依赖库（包括 Scala 标准库和我们使用的所有第三方库），因此第二个 Jar 包实际上和普通的 Java 程序导出的 Jar 包是一样的，可以像 Java 程序导出的 Jar 包一样直接执行：

  ```
  java -jar templateextractor_2.10-0.1-SNAPSHOT-one-jar.jar
  ```

- 导出 Eclipse 工程：为了使用 Eclipse 进行开发，需要将 SBT 工程导出为 Eclipse 工程。安装 SBT 的 sbteclipse 插件后，在 SBT 控制台输入 `eclipse` 即可导出 Eclipse 工程。

有一点需要注意，使用 sbteclipse 导出成 Eclipse 工程时，原来 SBT 工程中的资源文件目录不会加入到导出工程的 classpath 中。代码中使用了 typesafe 的 config 库管理和读取配置文件，配置文件 application.conf 放在 PROJECT_DIR/conf 下。为了使 config 库可以正确找到配置文件，需要在 PROJECT_DIR/build.sbt 中将 PROJECT_DIR/conf 加入到 SBT 工程的资源文件目录中，而使用 sbteclipse 导出时该目录将不会出现在导出的 Eclipse 工程的 classpath 中，此时如果直接运行导出后的 Eclipse 工程，config 库将无法找到配置文件。因此在使用 sbteclipse 导出 Eclipse 工程后，还需要在 Eclipse 中手动将 PROJECT_DIR/conf 目录加入到"Build Path" 中。

- 安装 SBT 插件：工程中安装了 sbt-onejar 和 sbteclipse 两个插件，使用 `addSbtPlugin` 函数安装插件，见 PROJECT_DIR/project 目录里的 build.sbt 文件（不是根目录 PROJECT_DIR 下的 build.sbt）。

2.3 一些 shell 脚本

从第 2.2节中可以看到，编译、运行一次代码需要很多步骤，手动操作的话很麻烦。另外，目前 Scala 编译器编译的速度比较慢，在实验室台式机上编译一次整个工程需要 3 分钟的时间（我的笔记本配置好一些，但也要 1 分钟左右）。因此，为了编译和调试代码方便，我写了一些 shell 脚本简化这些步骤，都放在了 PROJECT_DIR/tools 目录下。这些脚本本身都非常简单易懂，下面仅对主要的脚本进行说明：

- `rsync_source.sh`：Scala 编译器可以有效使用多个线程同时编译，因此放到服务器上可以大大加速编译速度。在服务器上的实测结果是编译整个工程只需要 10s 左右的时间。这个脚本的作用是利用 rsync 将全部代码同步到服务器上，以便在服务器上可以进行代码的编译。该脚本一般不直接使用，而是被其他的脚本调用。

- `make_jar.sh`：程序的开发和调试放在本地，运行则在服务器上，因此需要准备两个不同的配置文件。PROJECT_DIR/conf 目录下有两个文件：application.conf 和 application.conf.server，分别用于本地和服务器上的配置。config 库默认使用 application.conf 作为配置文件，在本地调试的时候这是没有问题的，如果要在服务器上运行时则需要更换配置文件。这个脚本的作用即是在导出成服务器上运行的 Jar 的时候设置 application.conf.server 为配置文件，并根据用户的输入自动修改 PROJECT_DIR/build.sbt 中类的路径，编译并打包成可运行的 Jar 包。这个脚本也不直接使用，而是被其他脚本调用。

- `send_server.sh`：这是主要使用的脚本。先调用 `rsync_source.sh` 自动将整个工程同步到服务器，然后调用 `make_jar.sh` 在服务器上编译并导出成 Jar 包，之后运行该 Jar 包。脚本接受一个参数，该参数为需要运行的类的路径。使用方法示例：
  ```
  tools/send_server.sh thu.ailab.template.TestTemplateManager
  ```

- `send_client.sh`：这个脚本和 `send_server.sh` 的作用和使用方法都一样，唯一的不同是这个脚本没有和服务器之间同步代码，而是在本地编译打包完后将最后导出的可运行的 Jar 包上传到服务器上。如果需要在本地进行编译，则使用这个脚本代替 `send_server.sh`。实际开发过程中我都是用 `send_server.sh`，没有使用这个脚本，因为本地的编译速度太慢了。

- `fix_template.sh`：这个脚本将服务器生成的模板 XML 文件转化为本地使用的模板 XML 文件。主要的作用是供另外一个工程 TemplateExtractorDemo 使用。

2.4 代码运行说明

之前已经介绍完了代码的管理，接下来说明如何运行程序。

程序分成多个模块，每个模块单独运行，前一个模块运行完了以后将结果输出成文本文件或者 XML，下一个模块读取这些文件恢复状态。所有的模块都可以通过 `send_server.sh` 来运行，比如预处理模块：

```
send_server.sh thu.ailab.document.TestDocProcessor
```

由于实验有几个不同的数据集，因此需要设定目前程序在那个数据集上运行，修改 PROJECT_DIR/conf 下的配置文件中的 `global.dataset` 即可，目前的取值可以为 `blog` 或 `news`。为了叙述方便，下面将用 `DATASET` 指代某个具体的数据集。

按顺序依次运行以下几个主要模块：

- 预处理：对应的类的路径为 `thu.ailab.document.TestDocProcessor`。运行该类后，原始的 HTML 文档将经过去除多余标签、遍历成序列、去除重复记录等步骤，最后，预处理后的文档将保存到另外一个目录中。
- 计算文档相似度：对应的类的路径为 `thu.ailab.distance.TestDocDistance`。运行该类后将计算所有输入文档两两之间的距离，并将距离保存到文本文件中。
- 文档聚类：对应的类的路径为 `thu.ailab.cluster.TestDocNaiveAggloCluster`。运行该类后将进行聚类，将聚类结果输出为 XML。
- 模板生成：修改配置文件 application.conf 中的 `global.stage`，将值改为 `build`，然后再用 `send_server.sh` 运行，对应的类的路径为 `thu.ailab.template.TestTemplateManager`。运行后将生成一个模板的 XML 文件。
- 模板标注：对应的类的路径仍为 `thu.ailab.template.TestTemplateManager`。同样需要修改配置文件 application.conf 中的 `global.stage`，将值改为 `extractRandom`。修改后运行程序，将随机抽取测试集中的某个样本，然后利用模板进行抽取。根据抽取的结果，决定我们真正需要的字段是哪些，然后对生成的模板 XML 文件中对应的部分进行标注。标注的类型在 thu/ailab/template/ExType.scala 中定义，可以根据需要往其中手动添加，目前有 `TITLE,CONTENT,AUTHOR` 三种，`MAGIC` 类型表示没有标注。自动生成的模板 XML 中默认所有的标签的 `exType` 值都为 `MAGIC`。下面举个例子：当我们决定某个 div 标签对应的为 `CONTENT` 时，我们修改 XML：

```
<treenode allowMultiple="false" exType="MAGIC">
      <names>
      <name>div</name>
      </names>
      <depths>
      <depth>9</depth>
      </depths>
</treenode>
```

中的 `exType` 属性，将其改为 `CONTENT`，如下所示：

```
<treenode allowMultiple="false" exType="CONTENT">
    <names>
    <name>div</name>
    </names>
    <depths>
    <depth>9</depth>
    </depths>
</treenode>
```

这样模板标注就完成了。

- 内容抽取：对应的类的路径还是 `thu.ailab.template.TestTemplateManager`。需要修改配置文件中的 `global.stage` 为 `extractAll`。从测试集中抽取的文件个数可以修改配置文件中的 `DATASET.template.extractCount`。抽取的结果将被保存为 XML 文件，文件路径由 `DATASET.template.extractResult` 决定。

2.5 配置文件说明

程序的正常运行非常依赖于配置文件。配置文件中的一些键值是程序运行的参数，如聚类的阈值，Actor 的个数等；另外很大一部分是程序输出文件路径的配置，如保存文档集合两两之间距离的文件路径，保存模板 XML 的文件路径等等，这里不一一列举说明，每个文件路径的具体含义可以参考代码中对应的使用到这些量的地方。文件路径中可以用 ~ 表示用户的家目录。

3 TemplateExtractorDemo 介绍

3.1 Play! 框架介绍

这个 Web Service 采用 Play! Framework 实现，网址为`playframework.org`，Play 的版本号为 2.1.1，该版本依赖的 Scala 版本为 2.10。Play 目前仍处于活跃开发中，不同版本有不兼容的情况，因此必须保证版本号一致。

Play 的工程底层采用 SBT 进行管理，使用方法上和 SBT 很类似。下载并配置好了 Play 之后，在命令行使用 `play` 进入到 Play 的控制台，这个控制台和 SBT 的控制台差不多，不过增加了一些命令。输入 `run` 即可在本地的 9000 端口运行我们的 Web Service。

3.2 Web Service 使用说明

下面使用 PROJECT_DIR 表示项目的根目录。为了正确运行程序，需要的文件有 TemplateExtractor 工程导出的 Templateextractor_2.10-0.1-SNAPSHOT-one-jar.jar 文件和 TemplateExtractor 工程依赖的包。可以使用第 2节中介绍的方法导出 templateextractor_2.10-0.1-SNAPSHOT-one-jar.jar，然后复制该 Jar 包到 PROJECT_DIR/lib 中，或者通过运行 PROJECT_DIR/tools/retrieve_txjar_local.sh 来自动完成编译、导出和复制的工作。TemplateExtractor 工程所依赖的包（jsoup，icu4j 等）也需要加入到 Play 工程的依赖中，依赖定义在文件 PROJECT_DIR/project/Build.scala 中。

Play 的配置文件为 PROJECT_DIR/conf/application.conf，需要向其中增加 `cluster.DocNaiveAggloCluster.clusterThreshold,templateFiles.blog,templateFiles.news` 三个值，后两者分别用于指定 blog 数据集和 news 数据集的标注好的模板文件的路径。

Web Service 的路径定义在 PROJECT_DIR/conf/routes 中，定义了 4 种路径，分别为：

```
/blog/l/*url
/news/l/*url
/blog/w/*url
/news/w/*url
```

/blog 和/news 前缀表示匹配那种数据集的模板，中间的 l 和 w 分别表示 local 和 web，最后的 *url 表示需要进行模板匹配的文件的 url。前两个中的 url 必须是本地路径，以 `/` 开头；后两个中的 url 必须是 Web 上的一个 url，以 `http://` 开头。前两个路径可以正常使用，只需要将 *url 部分替换为一个本地测试 HTML 文件的地址即可；而后两个路径暂时存在一些问题：由于目前用于训练的文件是之前的同学已经抓取好的网页，和 Web 上直接得到的有些不同，匹配时有一些差错，测试发现后两个路径暂不能正常使用。

需要注意的是，由于模板的 XML 文件是在服务器上生成的，而目前的 Web Service 则是在本地运行，因此需要将该模板 XML 中一些和路径相关的东西进行转化，即将服务器上的相关路径和文件转化为本地的路径和文件。在 TemplateExtractor 工程根目录下的 tools 目录里有一个 `fix_template.sh`shell 脚本负责这个转化。这个脚本接受一个参数，该参数为服务器生成的模板文件的路径，详细的实现可以参考该脚本的具体内容。

运行这个 Web Service 的步骤很简单，在 Play 控制台输入 `run` 之后，访问 localhost:9000 对应的路径即可。示例：在浏览器中访问 http://localhost:9000/blog/l//home/qjp/Programs/BachelorThesis/Data/blog1000/http%3A%2F%2Fblog.sina.com.cn%2Fs%2Fblog_000612600100y7ne.html，http://localhost:9000/blog/l/后面的 url 部分是我本地的某个测试文件的地址，可以得到


琴韵书声文库
http://blog.sina.com.cn/u/397920 [订阅] [手机订阅]
AUTHOR
首页 | 博文目录 | 图片 | 关于我
正文
字体大小： 大 中 小
电视散文《小桥·流水·人家》作者：杨健 叶兆君 (2012-03-11 10:58:58)
转 载
标签： 电视散文 小桥流水人家 杨健 叶兆君 央视国际 琴韵美文 文化 分类： 琴韵美文
小桥·流水·人家
作者：杨健 叶兆君
周庄，江苏省昆山市的一个古镇。四面环水，犹如泊在湖上的一片荷叶散发着淡淡的清香。 “ 吴水依依吴水流，吴中舟楫好夷游 ” ，小桥、流水、人家，千
书声
微博