MapReduce 课程设计 之

金庸的江湖

1　课程设计目标

通过一个综合数据分析案例："金庸的江湖——金庸武侠小说中的人物关系挖掘"，来学习和掌握 MapReduce 程序设计。通过本课程设计的学习，可以体会如何使用 MapReduce 完成一个综合性的数据挖掘任务，包括全流程的数据预处理、数据分析、数据后处理等。

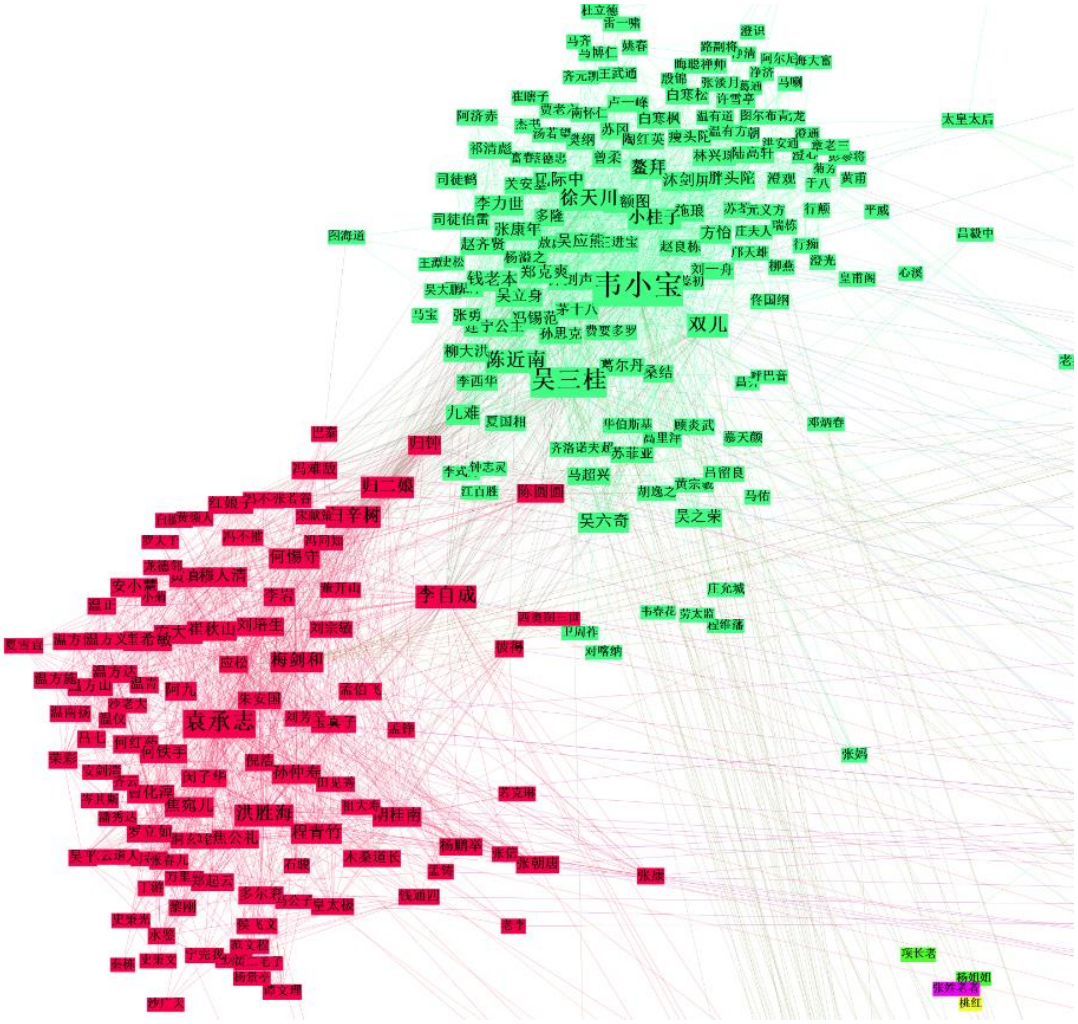

图 1 部分分析结果展示（碧血剑+鹿鼎记）。人物名字的大小由人物顶点的度数确定，人物标签的颜色根据标签传播算法的分析结果确定。

2 学习技能

通过本课程设计，可以熟悉和掌握以下 MapReduce 编程技能：

1. 在 Hadoop 中使用第三方的 Jar 包来辅助分析；
2. 掌握简单的 MapReduce 算法设计：
   a) 单词同现算法；
   b) 数据整理与归一化算法；
   c) 数据排序（选做）；
3. 掌握带有迭代特性的 MapReduce 算法设计：
   a) PageRank 算法；
   b) 标签传播（Label Propagation）算法（选做）。

3 任务描述

本课程设计包括如下的若干任务。这些任务组合起来，就构成了一个完整的人物关系分析流程。

任务 1 数据预处理

本任务的主要工作是从原始的金庸小说文本中，抽取出与人物互动相关的数据，而屏蔽掉与人物关系无关的文本内容，为后面的基于人物共现的分析做准备。

输入输出

数据输入：1.全本的金庸武侠小说文集（未分词）；2. 金庸武侠小说人名列表。

数据输出：分词后，仅保留人名的金庸武侠小说全集。

示例

输入：金庸 03 连城诀.txt 中的某一段内容

> **狄云**和**戚芳**一走到万家大宅之前，瞧见那高墙朱门、挂灯结彩的气派，心中都是暗自嘀咕。**戚芳**紧紧拉住了父亲的衣袖。**戚长发**正待向门公询问，忽见**卜垣**从门里出来，心中一喜，叫道：“卜贤侄，我来啦。”

输出：

> 狄云 戚芳 戚芳 戚长发 卜垣

任务 2 特征抽取：人物同现统计

本任务的重要完成基于单词同现算法的人物同现统计。在人物同现分析中，如果两个人在原文的同一段落中出现，则认为两个人发生了一次同现关系。我们需要对人物之间的同现关系次数进行统计，同现关系次数越多，则说明两人的关系越密切。

输入输出

输入：任务 1 的输出；

输出：在金庸的所有武侠小说中，人物之间的同现次数。

示例

输入：

狄云 戚芳 戚芳 戚长发 卜垣
戚芳 卜垣 卜垣

输出：

<狄云，戚芳> 1
<狄云，戚长发> 1
<狄云，卜垣> 1
<戚芳，狄云 > 1
<戚芳，戚长发 > 1
<戚芳，卜垣 > 2
<戚长发，狄云 > 1
<戚长发，戚芳 > 1
<戚长发，卜垣 > 1
< 卜垣，狄云> 1
< 卜垣，戚芳> 2
< 卜垣，戚长发> 1

任务 3 特征处理：人物关系图构建与特征归一化

当获取了人物之间的共现关系之后，我们就可以根据共现关系，生成人物之间的关系图了。人物关系图使用邻接表的形式表示，方便后面的 PageRank 计算。在人物关系图中，人物是顶点，人物之间的互动关系是边。人物之间的互动关系靠人物之间的共现关系确定。如果两个人之间具有共现关系，则两个人之间就具有一条边。两人之间的共现次数体现出两人关系的密切程度，反映到共现关系图上就是边的权重。边的权重越高则体现了两个人的关系越密切。

为了使后面的方便分析，还需要对共现次数进行归一化处理：将共现次数转换为共现概率，具体的过程见后面的示例。

输入输出

输入：任务 2 的输出

输出：归一化权重后的人物关系图

示例

输入：

<狄云，戚芳> 1
<狄云，戚长发> 1
<狄云，卜垣> 1
<戚芳，狄云 > 1
<戚芳，戚长发 > 1
<戚芳，卜垣 > 2
<戚长发，狄云 > 1
<戚长发，戚芳 > 1
<戚长发，卜垣 > 1
< 卜垣，狄云> 1
< 卜垣，戚芳> 2
< 卜垣，戚长发> 1

输出：

狄云 [戚芳,0.33333 | 戚长发，0.333333 | 卜垣 0.333333]
戚芳 [狄云,0.25 | 戚长发，0.25 | 卜垣 0.5]
戚长发 [狄云,0.33333 | 戚芳，0.333333 | 卜垣 0.333333]
卜垣 [狄云 0.25 | 戚芳,0.5 | 戚长发，0.25]

首先是将统计出的人物共现次数结果，转换成邻接表的形式表示：每一行表示一个邻接关系。

“戚芳 [狄云,0.25 | 戚长发，0.25 | 卜垣 0.5] ”表示了顶点“戚芳”，有三个邻接点，分别是“狄云”、“戚长发”和“卜垣”，对应三条邻接边，每条邻接边上分别具有不同的权重。这个邻接边的权重是依靠某个人与其他人共现的“概率”得到的，以“戚芳”为例，她分别与三个人（“狄云”共现 1 次、“戚长发”，共现 1 次、“卜垣”共现 2 次）有共现关系，则戚芳与三个人共现的“概率”分别为 1/(1+1+2) = 0.25，1/(1+1+2) = 0.25， 2/(1+1+2) = 0.5。这三个“概率”值对应与三条边的权重。通过这种归一化，我们确保了某个顶点的出边权重的和为 1。

任务 4 数据分析：基于人物关系图的 PageRank 计算

在给出人物关系图之后，我们就可以对人物关系图进行一个数据分析。其中一个典型的分析任务是：PageRank 值计算。通过计算 PageRank，我们就可以定量地金庸武侠江湖中的“主角”们是哪些。

输入输出

输入：任务 3 的输出

输出：人物的 PageRank 值

任务 5 数据分析：在人物关系图上的标签传播（选做）

标签传播（Label Propagation）是一种半监督的图分析算法，他能为图上的顶点打标签，进行图顶点的聚类分析，从而在一张类似社交网络图中完成社区发现（Community Detection）。图 1 中人物顶点的颜色就是根据标签传播的结果进行的染色。

参考资料

1. 英文资料：标签传播算法英文原始文献可参考原始英文论文中的 *III. COMMUNITY DETECTION USING LABEL PROPAGATION* 一节内容。

2. 中文资料：http://www.cnphp6.com/archives/24136

输入输出

输入：任务 3 的输出

输出：人物的标签信息。（可以直接将标签值输出，也可以使用 Gephi、Neo4j、ECharts 等工具进行数据可视化）

算法优化

针对本任务的需求，可以对标签传播算法进行一些优化，比如：在初始化阶段设置合适的"社区中心"节点，以减少迭代次数；对一些同时出现在多本书中的角色进行处理，防止他们的标签值互相干扰；设计合适的标签数据类型，以减少计算量和通信开销等。

可以从分析结果的准确性、计算用时、资源消耗、迭代次数等角度对优化措施的效果进行描述。

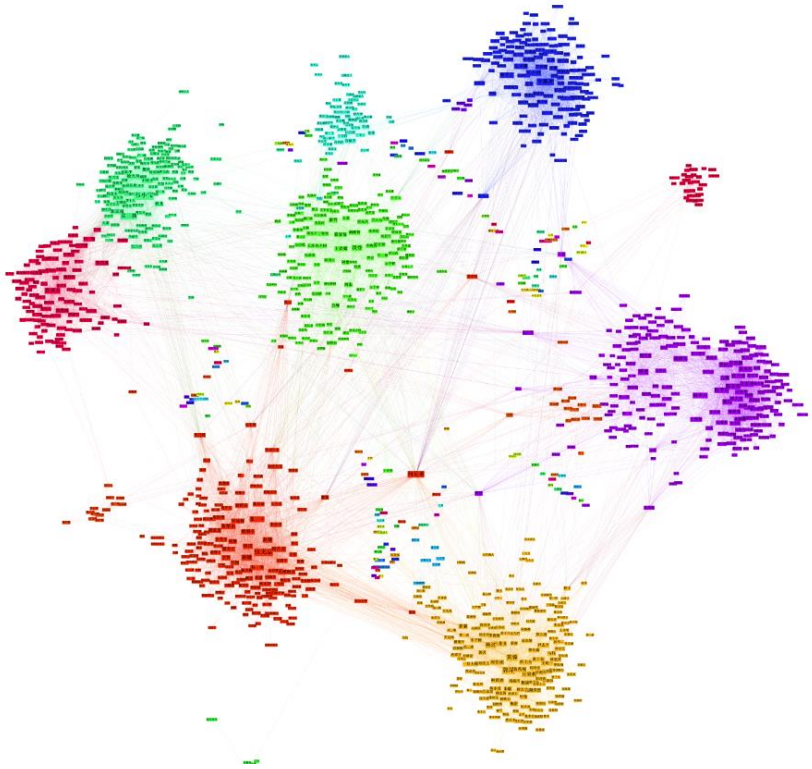

图 2 标签传播的结果展示

任务 6 分析结果整理（选做）

任务 4 和任务 5 默认的输出内容是杂乱的，从中无法直接的得到分析结论，因此我们需要对上述分析的结果进行整理、从而使人可以很容易地从分析结果中发现一些有趣的结论。

对于任务 5（PageRank）的输出，可以通过写一个全局排序的程序对人物的 PageRank 值进行排序，从而很容易地发现 PageRank 值最高的几个人物。排序工作可以通过一个排序 MapReduce 程序完成，也可以将 PageRank 值导入 Hive 中，然后利用 Hive 完成排序。

对于任务 6（标签传播）的输出，可以通过写一个 MapReduce 程序，将属于同一个标签的人物输出到一起，便于人来查看标签传播的结果。此外，在任务 6 的结果中，同一本书中的角色往往具有同样的标签，因此可以根据标签值与作品名的对应关系（根据数据集统计得出，也可以手动设置，存储在 job 的全局变量中），将所有人名按照其所属的小说分类输出。

4　提交材料

请各位同学提交如下材料。

1、程序源代码，要求提供包含完整目录结构的 src 代码包，并且提供编译方法说明。

2、程序可执行 jar 包以及 jar 包的执行方式。本题目的运行环境在 hadoop-2.7、jdk-1.7 环境下，必须采用 MapReduce 编程模型。

3、程序设计报告。报告内容包括程序设计的主要流程、程序采用的主要算法、进行的优化工作、优化取得的效果、程序的性能分析以及程序运行截图等。每个小组需要标明组内成员的分工情况。

实现指导

针对在课程设计的实现中可能遇到的问题，在这里整理了一份 FAQ，供参考。

关于任务 1

Q：如何对原始文本进行分词?

A：可以使用 Ansj_seg 工具。建议使用 3.2 版本。Ansj_seg 支持对中文文本进行分词，并且可以添加用户自定义的词典，这样它可以准确识别金庸武侠小说中的人名。

Q：如何在 Ansj_seg 中如何添加金庸小说人名，使它能正确分词人名?

A：请参考 Ansj_seg-动态添加页面的帮助指南。通过如下代码将金庸小说中的人名一个一个地添加到 Ansj_seg 的用户词典中。

```
UserDefineLibrary.insertWord(PersonName, "nr", 1000);
//"nr"表示这个词是人名
```

Q:具体地怎样用 Ansj_seg 进行分词?

A: 请参考 Ansj_seg-精确分词的帮助页面。

```
List<Term> parse = ToAnalysis.parse("让战士们过一个欢乐祥和的新春佳节。");
```

Ansj_seg 对一句话分词之后，每个词都使用 Term 对象来表示。Term 对象可以使用方法 getNatureStr 来获取每个词的词性，根据词性（包含"nr"的为人名），即可识别出人名。在 Map 阶段的输出中，只保留人名输出即可。

Q:如何在 Hadoop 的 Mapper 或 Reducer 中使用第三方的 Jar 包?

A：在打 Jar 包的时候，将第三方的依赖包也同步打入生成的 Jar 包。

- Eclipse:具体做法可以参考 Stack Overflow 上的问答。

- IntelliJ IDEA:在 IntelliJ IDEA 中，在创建 Artifacts 的时候，在 Output Layout 选项卡中，从右侧的 Available Elements 区域中，选中所要引入的第三方 Jar 包，在右键菜单中选择 Extract Into Output Root。

Q:如何将人名列表传入 Mapper/Reducer?

A: 这个问题可以转换成更通用的问题：如何在 Driver 端和 Mapper/Reducer 端之间传递数据。有三种方法。

1. 使用 Job 的 Configuration 在 Driver 与 Mapper 端进行传递。在 Driver 端的 main 函数中，从本地的文件中读入人名列表。将人名列表转化为一个长字符串。通过 Configuration 的 set 方法，将人名列表信息存入 Configuration 中。

   ```
   job.getConfiguration().set("OurNameList",人名列表长字符串);
   ```

   而在 Mapper/Reducer 端的 setup 函数中，可以通过代码

   ```
   context.getConfiguration().get("OurNameList")
   ```

   从 Configuration 中读取出刚才在 Driver 端设置的人名列表长字符串。通过这种方式，我们就实现了从 Driver 端到 Mapper/Reducer 端的数据传递。这种传递方式适合传递少量的数据。

2. 可以通过 Hadoop 提供的 Distributed Cache 机制传递数据。

3. 也可以通过 HDFS 传递数据。将人名列表存入一个 HDFS 的文件中。在 Mapper 的 setup 函数中，将名单列表从 HDFS 中读取到内存中。这种传递方式适合传递大规模的文件。

有关任务 4、5

Q：算法参数如何选取?

A：请自己尝试喽，不同参数会有不同的分析效果。