数据集：

来自于 Stack Overflow

下载地址是：https://archive.org/download/stackexchange/stackoverflow.com-Posts.7z

更新时间：10-Mar-2016 02:36

大小：8.3G

找到某个领域的专家的方式：

- 可以使用 Stack Overflow 中的口碑评分

  无法指定某个领域，比如 java。

  无法手动调节。
- 计算回答数量：

  无法度量答案的重要性
- 创建社交网络并计算用户的中心度

  PageRank，HITS

构建一个图：

- 节点：每个用户作为一个节点
- 边：一个问题的拥有者，指向一个被接收答案的拥有者。

方法概述：

1. 从输入中提取相关领域
2. 选择问题中包含某个领域的关键字，比如 Java
3. 通过找到被接受答案的拥有者创建图
4. 分析图

原始数据：

- 在 Posts.xml 文件中问题 Questions 的 XML 格式：

  <row Id=”4” PostTypeId=”1” OwnerUserId=”8” AcceptedAnswerId=”7” Tags=”<c#><winforms><forms><opacity>” .../>

Field	Value
Id	4
PostTypeId	1
OwnerUserId	8
AcceptedAnswerId	7
Tags	C#, winforms, forms, opacity

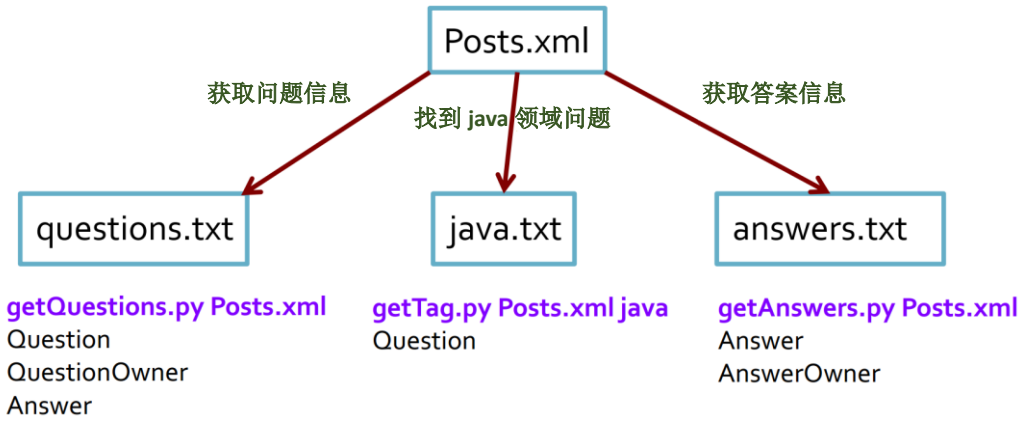
- 在 Posts.xml 文件中回答 Answers 的 XML 格式：

  <row Id=”12” PostTypeId=”2” OwnerUserId=”1” .../>

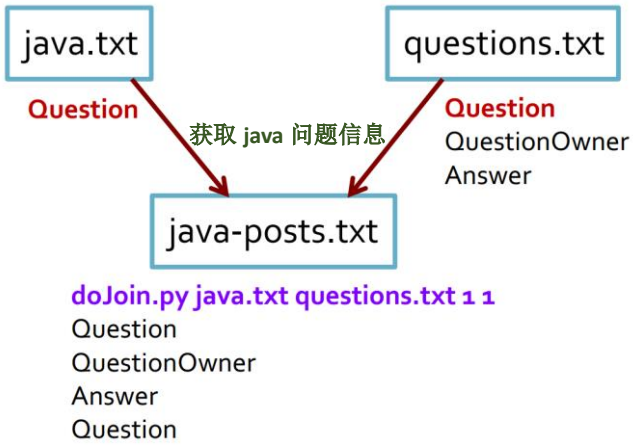
Field	Value
Id	12
PostTypeId	2
OwnerUserId	1

具体步骤：

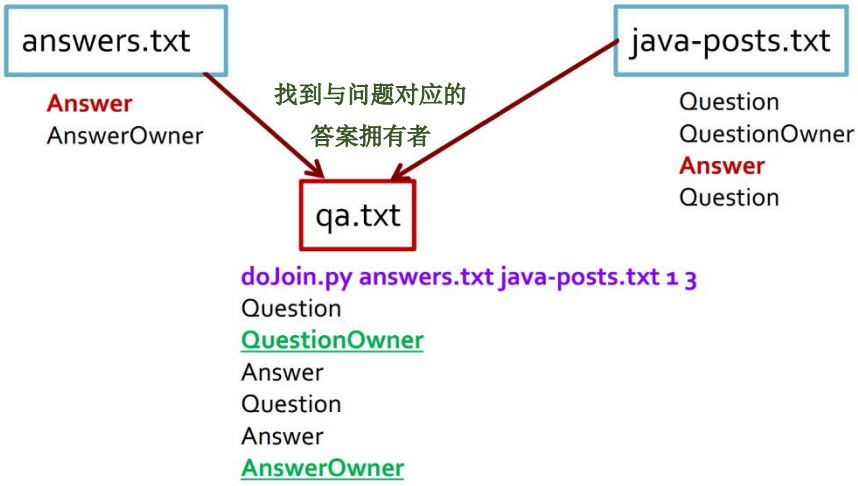
1. 处理输入文件，提取相关文件。获取问题和回答的列表，找到相关领域(如 java)的问题，共生成三个文件，question.txt，answers.txt， java.txt。



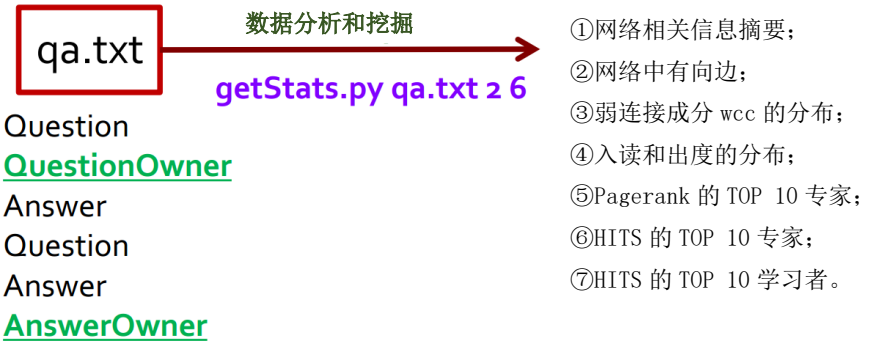
2. 选择只和相关领域(如 java)相关的问题：通过 java.txt 和 question.txt 合并出 java-posts.txt



3. 通过找到被接收答案的拥有者，并创建图：通过 answers.txt 和 java-posts.txt 合并出 qa.txt，其中所存放的节点对就是图的有向边。



4. 分析图：找到 TOP java 专家。



算法介绍：

1. Pagerank

PageRank 是 Google 专有的算法，用于衡量特定网页相对于搜索引擎索引中的其他网页而言的重要程度。它由 Larry Page 和 Sergey Brin 在 20 世纪 90 年代后期发明。PageRank 实现了将链接价值概念作为排名因素。

一个页面的“得票数”由所有链向它的页面的重要性来决定，到一个页面的超链接相当于对该页投一票。一个页面的 PageRank 是由所有链向它的页面（“链入页面”）的重要性经过递归算法得到的。一个有较多链入的页面会有较高的等级，相反如果一个页面没有任何链入页面，那么它没有等级。

假设一个由 4 个页面组成的小团体：A，B，C 和 D。如果所有页面都链向 A，那么 A 的 PR（PageRank）值将是 B，C 及 D 的 Pagerank 总和。

$$PR(A) = PR(B) + PR(C) + PR(D)$$

继续假设 B 也有链接到 C，并且 D 也有链接到包括 A 的 3 个页面。一个页面不能投票 2 次。所以 B 给每个页面半票。同样，D 投出的票只有三分之一算到了 A 的 PageRank 上。

$$PR(A) = \frac{PR(B)}{2} + \frac{PR(C)}{1} + \frac{PR(D)}{3}$$

换句话说，根据链出总数平分一个页面的 PR 值。

$$PR(A)=\frac{PR(B)}{L(B)}+\frac{PR(C)}{L(C)}+\frac{PR(D)}{L(D)}$$

最后，所有这些被换算为一个百分比再乘上一个系数。由于“没有向外链接的页面”传递出去的 PageRank 会是 0，所以，Google 通过数学系统给了每个页面一个最小值：

$$PR(A)=\left(\frac{PR(B)}{L(B)}+\frac{PR(C)}{L(C)}+\frac{PR(D)}{L(D)}+\cdots\cdots\right)d+\frac{1-d}{N}$$

说明：在 Sergey Brin 和 Lawrence Page 的 1998 年原文中给每一个页面设定的最小值是 1-d，而不是这里的(1-d)/N。所以一个页面的 PageRank 是由其他页面的 PageRank 计算得到。Google 不断的重复计算每个页面的 PageRank。如果给每个页面一个随机 PageRank 值（非 0），那么经过不断的重复计算，这些页面的 PR 值会趋向于稳定，也就是收敛的状态。

2. HITS

HITS 算法是由康奈尔大学的 Jon Kleinberg 博士于 1997 年首先提出的,为 IBM 公司阿尔马登研究中心的名为“CLEVER”的研究项目中的一部分。

按照 HITS 算法，用户输入关键词后，算法对返回的匹配页面计算两种值，一种是枢纽值（Hub Scores），另一种是权威值(Authority Scores),这两种值是互相依存、互相影响的。所谓枢纽值，指的是页面上所有导出链接指向页面的权威值之和。权威值是指所有导入链接所在的页面中枢纽之和。

具体算法如下:

将查询 q 提交给基于关键字查询的检索系统，从返回结果页面的集合中取前 n 个网页(如 n=200)，作为根集合(root set)，记为 S，则 S 满足：

1.S 中的网页数量较少

2.S 中的网页是与查询 q 相关的网页

3.S 中的网页包含较多的权威(Authority)网页

通过向 S 中加入被 S 引用的网页和引用 S 的网页,将 S 扩展成一个更大的集合 T. 以 T 中的 Hub 网页为顶点集 V1 ,以权威网页为顶点集 V2 。

V1 中的网页到 V2 中的网页的超链接为边集 E ,形成一个二分有向图. 对 V1 中的任一个顶点 v ,用 h (v) 表示网页 v 的 Hub 值,且 h (v)收敛;对 V2 中的顶点 u ,用 a (u) 表示网页的 Authority 值。

开始时 h (v) = a (u) = 1 ,对 u 执行 I 操作,修改它的 a (u) ,对 v 执行 O 操作,修改它的 h (v) ,然后规范化 a (u),h (v) ,如此不断的重复计算下面的 I 操作和 O 操作,直到 a (u),h(v)收敛 。

其中 I 操作:a (u) = Σh (v) ;O 操作: h (v) = Σa (u) 。每次迭代对 a (u) 、h (v) 进行规范化处理: a (u) = a (u)/Σ[a (q)]2 ; h (v) = h (v)/Σ[h (q)]2 。