

# Wikipedia Index Construction

—Course Project for Distributed Systems

何东 15307130195

## 1. 项目概述

本次项目我选择的是建立 Wikipedia 索引。利用在 Hadoop 集群上的 MapReduce，我做了如下的工作：1) 首先计算出每个单词 (term) 在每篇文章中的 TF (Term Frequency)；2) 利用计算好的 TF 结果计算出每个单词 (term) 的 DF (Document Frequency)；3) 利用上述计算好的 TF 和 DF 结果计算出每个单词 (term) 在每篇文章中的 TFIDF (term frequency-inverse document frequency)；4) 对于每篇文章，利用计算好的 TFIDF 结果求出 TFIDF 最大的五个单词，用于之后的 Wikipedia 查询服务；5) 计算出每个 Wikipedia 页面 (page) 在原始 Wikipedia XML 中的偏移 (offset) 和长度 (length)；6) 用 Python Flask 框架搭建服务器，利用 4) 和 5) 计算出的结果，实现 Wikipedia 的关键词搜索功能，并且支持多关键词查询。前五个任务我都设计并实现了 MapReduce 任务，并且在 Hadoop 集群上成功运行得到结果。

## 2. 实现思路与细节

### 2.1. 计算 TF (TermFrequencyCalculator.java)

计算 TF 即计算每个单词在每篇文章中出现的次数。由于 MapReduce 中的 Mapper 默认为按行进行 map，我需要将其转换为基于<page></page>的标签进行 map。参考网上资料，我实现了继承 TextInputFormat 的 XmlInputFormat 用作 Mapper 的输入。此任务输入文件为原始的 Wikipedia XML 文件。

**Map 过程：**输入：key 为 page 在原始 XML 中的偏移 (offset)；value 为 page 的内容；输出：key 为页面序号 (pageId) 与单词 (term) 组成的 Text，即“pageId#term”；value 为一个二元组，即 (文章中的单词总数, 1)。

大体过程为提取 page 中<text>与</text>中间的内容，去除非英文字符后统计单词数量；然后对每一个单词输出 (key, value)。

**Reduce 过程：**输入：key 为页面序号 (pageId) 与单词 (term) 组成的 Text，即“pageId#term”；value 为一个二元组，即 (文章中的单词总数, 1)；输出：key 为页面序号 (pageId) 与单词 (term) 组成的 Text，即“pageId#term”；value 为一个实数，代表一个单词 (term) 在这个页面 (page) 中的 TF 值 (Term Frequency)。

大体过程为累加输入中的 value 二元组中的第二项，然后用这个累加的和除以文章的单词总数获得 TF 值。

### 2.2. 计算 DF (DocumentFrequencyCalculator.java)

计算 DF 即计算每个单词总共在多少个页面（page）中出现过，这可以利用已经算出的 TF 值计算得到。此任务输入文件为上面得到的 TF 文件。

**Map 过程：**输入：key 为当前行在输入文件中的偏移（offset）；value 为“pageId#term TermFrequency”；输出：key 为单词（term）；value 为一个整数，值为 1。

**Reduce 过程：**输入：key 为单词（term）；value 为一个整数，值为 1；输出：key 为单词（term）；value 为一个整数，代表一个单词（term）的 DF。

这么做的原理是，在 TF 的计算结果中，同一篇文章中的同一个单词只会出现一次。那么，我只需统计 TF 的计算结果中相同的单词（term）出现了多少次。

## 2.3. 计算 TFIDF（TFIDFCalculator.java）

TFIDF 可以利用已经算出的 TF 值和 DF 值计算得到。此任务输入文件为上面得到的 TF 文件和 DF 文件。

**Map 过程：**输入：key 为当前输入行的偏移（offset）；value 为 TF 文件中的一行或是 DF 文件中的一行；输出：key 为单词（term）；value：1）若 map 的输入为 TF 文件的一行，value 为页面编号（pageId）与单词 TF 值组成的二元组，即（pageId, TF），2）若 map 的输入为 DF 文件的一行，则 value 为由 DF 值计算得到的 IDF（Inverse Document Frequency）值。

**Reduce 过程：**输入：key 为单词（term）；value 为，1）页面编号（pageId）与单词 TF 值组成的二元组，即（pageId, TF），2）或是单词 IDF 值；输出：key 为页面序号（pageId）与单词（term）组成的 Text，即“pageId#term”；value 为单词的 TFIDF 值。

该过程通过 Reduce 把同一个单词（term）的 TF 值和 DF 值联系到一起，并由此算出了每篇页面（page）中的每个单词（term）的 TFIDF 值。由于 Reduce 的输入有可能是（pageId, TF），也有可能是 IDF，并且我需要先知道这个单词（term）的 IDF 才能计算 TFIDF 值，所以我要对 Reduce 输入的 values 进行两次遍历，因为我们不知道 IDF 在 values 中的哪个位置。而又因为 values 是 Iterable 类型（遍历过一次之后无法找回），所以，在进行第一次遍历 values 的时候，要做一个 values 的备份，以便第二次遍历。

## 2.4. 为每个页面计算 TFIDF 最大的五个单词（TFIDFSelector.java）

为了加速后面的搜索服务，我为每个页面（page）计算出 TFIDF 值最大的五个单词（term），以此代表该页面参与搜索。此任务输入文件为上面得到的 TFIDF 文件。

**Map 过程：**输入：key 为当前输入行的偏移（offset）；value 为 TFIDF 文件中的一行，即为“pageId#term TFIDF”；输出：key 为页面编号（pageId）；value 为单词（term）与它的 TFIDF 组成的二元组，即为（term, TFIDF）。

**Reduce 过程：**输入：key 为页面编号（pageId）；value 为单词（term）与它的 TFIDF 组成的二元组，即为（term, TFIDF）；输出：key 为页面序号（pageId）；value 为该页面中拥有最大的 TFIDF 值的五个单词（term）。

## 2.5. 计算页面的偏移和长度（PageOffsetAndLengthCalculator.java）

为了加速后面的搜索服务，我为每个页面（page）计算出了它在原始 XML 文件中的偏移（offset）和长度（length）。此任务输入为原始的 Wikipedia XML 文件。

**Map 过程：**输入：key 为 page 在原始 XML 中的偏移（offset）；value 为 page 的内容；输出：key 为页面编号（pageId）；value 为页面的偏移（offset）和长度（length）。

**Reduce 过程：**无需。

## 2.6. Web 搜索服务



我用 Python Flask 搭建了一个服务器，实现了关键词搜索的功能，并且实现多关键词搜索，并展示原始 Wikipedia 页面。

其工作流程为：

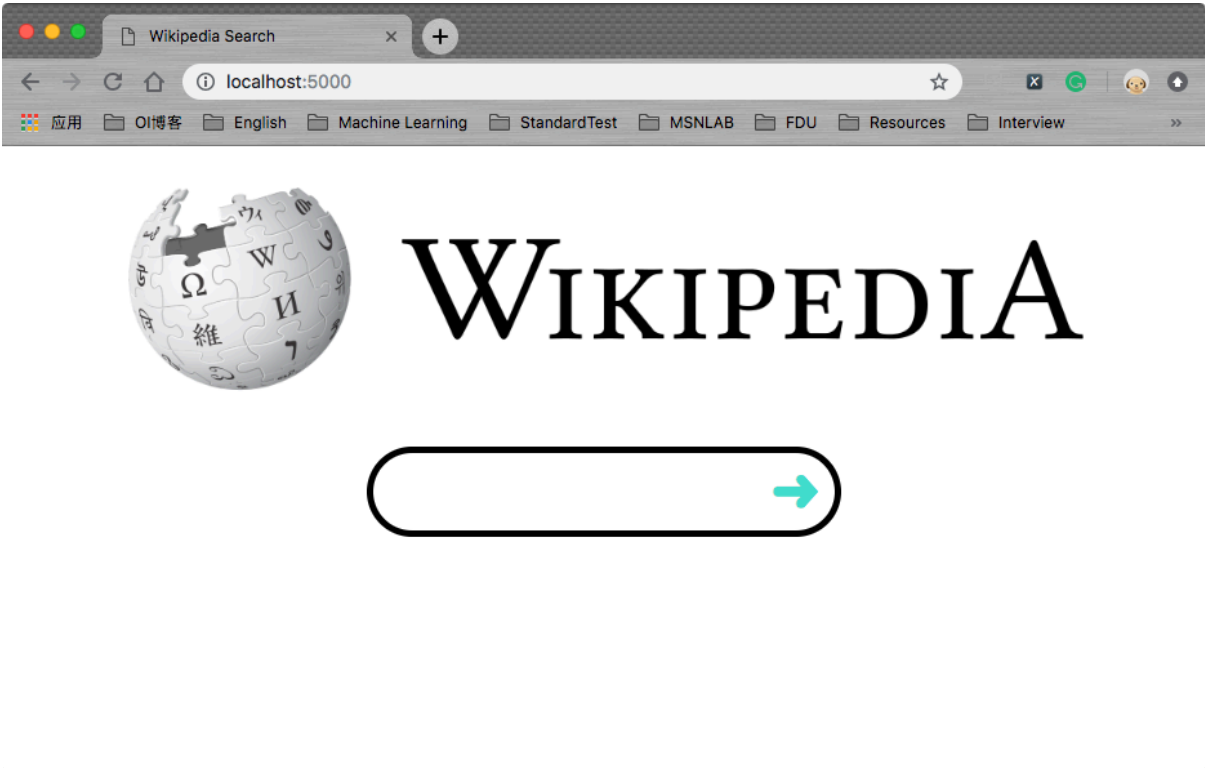
0. 启动服务器，将页面的偏移和长度以及 TFIDF 最大的五个单词读入内存，并用 dict（Hash 表）存储。
1. 在浏览器输入查询的单词（或词组）后，服务器用查询单词与内存中的单词比对，得到候选页面编号。
2. 服务器利用候选页面编号查询到页面的偏移和长度，并在原始文件中获取页面的数据，将候选页面的标题显示到搜索结果中。
3. 当用户点击搜索结果的页面链接，服务器通过页面偏移和长度拿到页面数据，并用 wikimarkup 库将 wiki 语法转为 html 页面展示给用户。

## 3. 项目展示

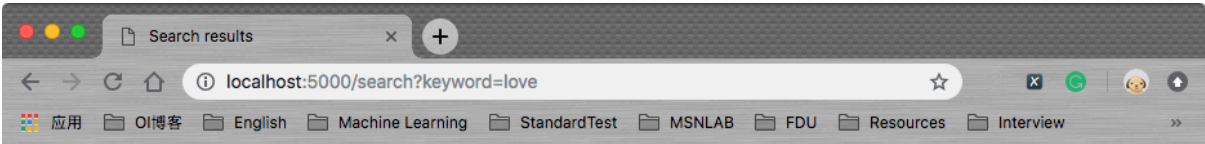
### 3.1. Web 搜索服务用到的部分 MapReduce 运行结果

名称	修改日期	大小	种类
 results_pageoffset	今天 下午2:44	49.4 MB	文本编辑 文稿
 results_selected_tfidf	今天 下午6:51	157.3 MB	文本编辑 文稿

3.2. Web 搜索界面



3.3. 搜索结果样例



Search Results

Page ID	Title
<a href="#">28</a>	A Letter to a Hindu
<a href="#">35</a>	The Sonnets
<a href="#">60</a>	The Phoenix and the Turtle
<a href="#">62</a>	The Passionate Pilgrim
<a href="#">70</a>	Sonnets to Sundry Notes of Music
<a href="#">252</a>	The Passionate Shepherd to His Love
<a href="#">286</a>	The Failures
<a href="#">857</a>	Love Songs
<a href="#">1630</a>	The First Kiss of Love
<a href="#">1736</a>	Serenade (Poe)
<a href="#">1945</a>	Leaves of Grass/Book IV
<a href="#">1991</a>	To the Willow-Tree
<a href="#">1996</a>	To His Coy Mistress