Web Search
万维网搜索

Web Search Engines
万维网搜索引擎

Notes based on two papers by David Hawking,

基于David Hawking的两篇论文的笔记，

Commonwealth Scientific and Industrial Research Organisation (CSIRO), Australia.

澳大利亚联邦科学与工业研究组织（CSIRO）。

the papers were written a few years ago but they are very informative still.

这些论文是几年前写的，但是他们仍然非常翔实。

Figures shown are approximate as they are increasing all the time and also hard to measure exactly.

所示的数字是近似的，因为它们一直在增加并且也难以精确测量。

The Scale of the Web
万维网有多大？

How many web pages?

有多少个网页？

Very hard to measure

很难衡量

1995	10,000,000 pages
2006	10,000,000,000 pages
2009	25,000,000,000 pages
2014	48,000,000,000 pages
2019	60,000,000,000 pages

http://www.worldwidewebsize.com/

The Problem
问题

60,000,000,000 pages	60,000,000,000页
Hundreds of languages	数百种语言
1,000,000,000 queries each day	每天1,000,000,000个查询
The need to respond to a query in less than one second	需要在不到一秒的时间内响应查询
How to do this?	我们怎样才能解决这个问题？

What to Index
索引什么

Not everything

不是一切

Should reject automated content if it is of low value e.g. time of day

如果自动内容价值较低，则应拒绝自动内容，例如一天中的时间

Data (e.g. daily measurements of ocean temperatures or astrophysical observations) should not be indexed

不应对数据（例如海洋温度或天体物理观测的每日测量）编制索引

Inaccessible pages (e.g. corporate intranets) can of course not be indexed

无法访问的页面（例如公司内部网）当然不能编入索引

Pages created in response to a query may not be accessible

可能无法访问为响应查询而创建的页面

Search Engine Infrastructure
搜索引擎的结构

Crawling & Indexing	Query processing	抓取和索引	查询处理
Gathering	Querying	搜集	查询
Extracting	Ranking	提取	排行
Indexing	Presenting	索引	呈现

However, engine must be highly parallel to achieve desired peformance

但是，发动机必须高度平行才能达到理想的性能

Engines use multiple data centres distributed around the world

引擎使用分布在世界各地的多个数据中心

Around 100,000 standard PCs are used to implement a search engine

大约100,000台标准PC用于实现搜索引擎

Web Data Centre
数据中心

Groups of PCs in a data centre are dedicated to different functions:

1. Crawling
2. Indexing
3. Query processing
4. Snippet generation
5. Link graph computations (i.e. connections between pages to estimate importance of a page)
6. Result caching (to avoid re-computation of results when the same query is repeated)
7. Insertion of advertising

数据中心中的PC组专用于不同的功能：

爬行
索引
查询处理
片段生成
链接图计算（即页面之间的连接以估计页面的重要性）
结果缓存（以避免在重复相同查询时重新计算结果）
插入广告

Replication
并行处理

Required to achieve high performance. We need:

要求实现高性能。我们需要：

- Many data centres

  许多数据中心

- Very fast network links between them

  它们之间的网络链接非常快

- A means of dividing tasks among data centres

  一种在数据中心之间划分任务的方法

Scale of Indexing and Query Processing
索引和查询处理的规模

400,000,000,000,000 **bytes** are crawled and indexed	将对400,000,000,000,000个字节进行爬网和编制索引
Index size is about 100,000,000,000,000 bytes (25% of collection size)	索引大小约为100,000,000,000,000字节（收集大小的25％）
Possibly 1,000,000,000 queries a day (2,000 per second equates to 172,800,000 queries a day)	每天可能有1,000,000,000个查询（每秒2,000个，相当于每天172,800,000个查询）
These figures are a few years ago...	这些数字是几年前的......

Web Crawling
网络爬行

1. Want to visit all the websites on the internet
2. Then, index them
3. Now, users can search

In our computer lab (e.g. Wikipedia), we have a list of the files

On the internet, we have no list...

We must search for files before we can index them

Call this ‘Crawling’.

想要访问互联网上的所有网站

然后，索引他们

现在，用户可以搜索

在我们的计算机实验室（例如维基百科）中，我们有一个文件列表
在互联网上，我们没有清单......

我们必须先搜索文件，然后再将它们编入索引
称之为'爬行'。

How to Start Crawling the Web
如何开始抓取网页

Have queue of URLs to be visited

Need means of telling if URL is already searched - a list of them could be 20,000,000,000 URLs!

Queue is seeded (i.e. initialised) with some 'good' sites e.g.

- https://dmoz-odp.org/ (was hand-built web directory until 2017)
- wikipedia.org, baike.com

1. High quality sites
2. Contain many links

有要访问的URL队列

需要告知URL是否已被搜索 - 它们的列表可能是20,000,000,000个URL！

队列是用一些“好的”站点播种（即初始化），例如

dmoz-odp.org
（手工制作的网页目录）
wikipedia.org, baike.com

高品质的网站
包含许多链接

Web Crawling Algorithm
网络爬行算法

1. Fetch the first page
2. Add links to the end of the queue if:
   1. not in the queue
   2. not visited already
3. Save page content for indexing
4. Remove URL from the queue
5. Continue until the queue is empty

获取第一页

在以下情况下添加指向队列末尾的链接：
不在队列中
尚未访问过
保存页面内容以进行索引

从队列中删除URL

继续，直到队列为空

Refinements to the Algorithm - Speed
算法的改进 - 速度

Can only be achieved through parallelism

可以通过并行实现

Assuming 1 second response from the average site (often slower), 86,400 pages per day is possible

假设1秒响应，每天86,400页

For 20,000,000,000 pages, 634 years will be needed!!

对于20,000,000,000页，将需要634年！

So, need multiple queries working in parallel

因此，需要并行处理多个查询

A *hash function* can be used: (see earlier discussion on hash functions)

可以使用哈希函数：

- Divide URLs into n categories
- Pass to n different crawlers

将网址划分为n个类别
传递给n个不同的抓取工具

So each URL is handled exactly once

因此，每个URL只处理一次

Refinements - Avoiding Site Bombardment
改进 - 避免过度访问网站

Overloading of sites must be avoided	必须避免重载站点
One request at a time to a server when crawling it	在抓取服务器时，一次向服务器发出一个请求
Gap between requests	请求之间的差距
Internet bottlenecks must not be overloaded	互联网瓶颈不得超载
Otherwise whole countries can be unable to access internet during crawling!	一个国家/地区在抓取过程中无法访问互联网！
File robots.txt says if a site does not wish to be indexed and this should be respected	文件robots.txt表示网站是否不希望编入索引

Duplicate Files
重复文件

Software documentation, Wikipedia files etc etc may be mirrored in several places

软件文档，维基百科文件等可能会在几个地方被镜像

However, they should not be indexed multiple times as this is inefficient

但是，它们不应被多次索引，因为这是低效的

Simple file comparisons can detect some duplicates

简单的文件比较可以检测一些重复

More sophisticated methods may be needed based on content

基于内容可能需要更复杂的方法

Duplicate Files
重复文件

Duplicates can have links to more duplicates... resulting in inefficiency

重复可以链接到更多重复...导致效率低下

Links can be relative and so when converted to absolute form are not the same as the original, and so may not be detected

链接可以是相对的，因此当转换为绝对形式与原始形式不同时，可能无法检测到

For URLs, remove unnecessary parameters e.g. session IDs and also convert to lower case before checking for duplication

对于网址，请删除不必要的参数会话ID，并在检查重复之前转换为小写

URLS and Upper Case/Lower Case
URLS和大写/小写

Consider http://en.example.org/WIKI/URL

Domain name (http://en.example.org) is not case sensitive

Path (/WIKI/URL) is case sensitive

Can depend on Operating System of Web Server...

- Linux distinguishes case - WIKI /= wiki
- Windows does not - WIKI == wiki

考虑http://en.example.org/WIKI/URL

域名（http://en.example.org）不区分大小写

路径（/ WIKI / URL）区分大小写

可以依赖Web服务器的操作系统......

Linux区别于案例 - WIKI / = wiki

Windows没有 - WIKI == wiki

How Often to Crawl?
您多久访问一次页面?

Same pages change rapidly - e.g. newspapers

相同的页面快速变化 - 例如报纸

Some pages never change e.g. archives, for example company announcements

有些页面永远不会改变，例如公司公告

How Often to Crawl?
您多久访问一次页面?

Assess priority of URL for crawling by:

- Frequency with which it changes
- Incoming link count (how many pages link to it)
- Click frequency (how often people go to the page)

Place URLs in the queue in priority order

评估用于抓取的网址的优先级：

变化的频率

传入链接计数（链接到多少页）

点击频率（人们访问该页面的频率）

按优先级顺序将URL放入队列中

Misleading ‘Spam’ Pages
误导或欺骗页面

These are pages which are designed to boost the rating of a site

这些页面旨在提高网站的评级

For example, they try to boost the link count of a site by creating new websites which link to the target website!

例如，他们尝试通过创建链接到目标网站的新网站来增加网站的链接数量！

Also, some sites try to deliver different content to crawlers than to users (called 'cloaking')

此外，一些网站尝试向抓取工具提供与向用户不同的内容（称为“隐藏真实内容”）

Search engine companies try to identify Spam sites and those they are inflating

搜索引擎公司试图识别垃圾邮件站点和他们正在膨胀的站点

Both are then punished by not being indexed

然后两者都没有编入索引

Documents to be Indexed by a Crawler
要由爬虫索引的文档

Not just HTML and XHTML files	不只是HTML和XHTML文件
XML files	XML文件
.pdf, .doc, .ps files etc.	.pdf，.doc，.ps等文件
Need filters for all of these which will extract the Ascii text from them so that it can be indexed	需要对所有这些过滤器进行过滤，这些过滤器将从中提取Ascii文本，以便对其进行索引
Filters may not be accurate. For example, extraction of text from .ps files is, surprisingly, not able to recover the exact sequence of Ascii characters	过滤器可能不准确。例如，令人惊讶的是，从.ps文件中提取文本无法恢复Ascii字符的确切序列

Conversion of Character Sets
字符集的转换

Many different encodings:

- Ascii
- GB2312
- SJIS
- Iso-Latin-1
- etc.

Convert all character sets to Unicode

So, all comparisons are Unicode-to-Unicode

许多不同的编码：

ASCII
GB2312
SJIS
ISO-LATIN-1
等等

将所有字符集转换为Unicode

因此，所有比较都是Unicode到Unicode

The Scale of Web Indexing
Web索引的规模

500 terms in 20,000,000,000 pages would result in a list of 10,000,000,000,000 entries during the first stage of indexing!

在索引的第一阶段，20,000,000,000页中的500个术语将导致10,000,000,000,000个条目的列表！

Speeding up:

加速中：

- Can divide URLs to be indexed between many machines

可以将URL划分为多台机器之间的索引

- e.g. use 400 machines to index 20,000,000,000 pages, i.e. 50,000,000 pages each

例如使用400台机器索引20,000,000,000页，即每页50,000,000页

- Now the initial file contains around 25,000,000,000 entries

现在，初始文件包含大约25,000,000,000个条目

Speeding Up cont.
加速中

The indexing process can be speeded up by building a partial inverted index in memory as documents are scanned

当扫描文档时，可以通过在存储器中构建部分反向索引来加速索引过程

When memory is used up, the partial inverted index is saved on disk and a new one commenced in memory

当内存用完时，部分反向索引保存在磁盘上，新的内容开始在内存中

Finally when all documents are scanned, the indexes are merged

最后，当扫描所有文档时，合并索引

Compression
数据压缩

Compression algorithms can be applied to indexing datastructures so that the space occupied by them on disk is reduced

压缩算法可以应用于索引数据结构，以便减少它们在磁盘上占用的空间

With some algorithms, the input can be decompressed on-the-fly as bytes are read

使用某些算法，可以在读取字节时即时解压缩输入

The time taken to decompress in memory is much less than the time taken to read information in from disk

在内存中解压缩所花费的时间远远少于从磁盘读取信息所花费的时间

Anchor Text
超链接文本

Link anchor text is the text which is highlighted at an anchor point. You can click on it.

链接锚文本是在锚点处突出显示的文本。你可以点击它。

Anchor text is indexed

- as part of the document
- also, as part of the document which it is pointing to

锚文本已编入索引
作为文件的一部分
此外，作为它所指向的文件的一部分

Anchor text indicates what target document is actually.

锚文本指示实际的目标文档。

Thus a good source of index terms for target document

因此，目标文档的索引术语的良好来源

Link Popularity Score
链接流行度分数

The number of pages which are linking to a document - the more pages pointing to it, the more important that page is

链接到文档的页面数量 - 指向它的页面越多，页面就越重要

The simplest way of calculating it is a count of the links

计算它的最简单方法是链接计数

PageRank takes into account

PageRank考虑到了

- How many links there are

  有多少链接

- How important the link is

  链接有多重要

PageRank Algorithm
PageRank算法

PageRank computes the importance of a webpage based on two factors: the pages which point to it and the links going from it

PageRank基于两个因素计算网页的重要性：指向它的页面和从中指向的链接

PageRank is then used as one of several factors in ranking the results

然后将PageRank用作排名结果的几个因素之一

$$PR(A) = (1-d) + d(PR(T_1)/C(T_1) + \dots + PR(T_n)/C(T_n))$$

Page A has pages T_1-T_n which point to it (i.e. cite it)

页面A有T1-Tn页面指向它（即引用它）

$C(T_i)$ is the number of links going out of page T_i

C（Ti）是从页面Ti出来的链接数

d is a damping factor, usually set to 0.85

d是阻尼系数，通常设为0.85

Points about PageRank
关于PageRank的要点

The PageRank varies with the number of pages pointing to page A – the more pages the better

PageRank随指向页面A的页面数量而变化 - 页面越多越好

The contribution of each pointing page varies with

- its PageRank (the higher, the more contribution)
- its 'out' count C (the lower, the more contribution

每个指向页面的贡献随着不同而变化
它的PageRank（越高，贡献越多）
它的'出'计数C（越低，贡献越大

The PageRank of a page depends on the PageRanks of all pages from which it can be reached by any number of steps

页面的PageRank取决于所有页面的PageRank，可以通过任意数量的步骤访问它

Computing PageRank Matrix

To compute the PageRank matrix:

- Set initial PageRank of all pages to 1-d, i.e. 0.15
- Go through all pages, computing PageRank using formula

Repeat the whole process many times

PageRank values will gradually spread through the network

Eventually, PageRank scores of all pages will stabilise

It is a complex task computationally

没有指向的页面的PageRank是1-d，即常数

要计算页面的PageRank：
直到没有任何意义的页面
向各个方向做，向后连接

计算Web上所有页面的PageRank涉及使用整个连接矩阵。

这是一项复杂的计算任务

Points about PageRank
关于PageRank的要点

A page has a high PageRank if:
- many pages point to it (of low/medium PageRank)
- few pages point to it (of high PageRank)

如果出现以下情况，页面的PageRank很高：
很多页面指向它（低/中PageRank）
几页指向它（高PageRank）

PageRank is very important for document ranking

PageRank对于文档排名非常重要

HostRanks uses host-host connectivity (not page-page)
- simpler computationally
- can assign same PageRank to all pages on a site

HostRanks使用主机 - 主机连接（不是页面）
计算简单
可以将相同的PageRank分配给站点上的al
页面

However, some pages on a site are more important than others

但是，网站上的某些页面比其他页面更重
要

Query Independent Score
查询独立分数

Can rank pages independently of any query:

- PageRank
- URL brevity (shorter URL is better)
- Click frequency

Can combine score based on these with the usual match of query with document (TF*IDF etc)

可以独立于任何查询排名页面:

网页排名
网址简洁（较短的网址更好）
点击频率

可以将基于这些的得分与查询与文档（TF * IDF等）的通常匹配相结合

Query Dependent Score
查询相关分数

TF and IDF as previously discussed earlier

Capitalisation - Capitalised terms score more highly

HTML heading size:

- All headings more important than normal text
- Larger headings more important than smaller ones

Italic or Bold font - these terms score more highly

TF和IDF如前所述

资本化 - 资本化条款得分更高

HTML标题大小：

所有标题比普通文本更重要

较大的标题比较小标题更重要

斜体或粗体字 - 这些术语得分更高

Query Dependent Score cont.
查询相关分数

Closeness of search terms in document - the closer they are, the higher the score

文档中搜索词的紧密度 - 它们越接近，得分越高

Position of terms in document - terms near the start may be more important

术语在文档中的位置 - 开头附近的术语可能更重要

Anchor text:

主播文字：

- indicates content of doc pointed to
- especially useful for images

表示doc指向的内容
对图像特别有用

Query Processing (Single Processor)
查询处理（单处理器）

Web queries tend to be very short

Web查询往往很短

Average query length is about 2.3 words

平均查询长度约为2.3个字

Most engines return docs containing all the query words:

大多数引擎返回包含所有查询词的文档：

- get postings lists for all query words and compute intersection
- terminate search when enough docs are found (e.g. 10)

获取所有查询词和计算交集的发布列表

找到足够的文档时终止搜索（例如10）

Query Processing (Multiple Processor)
查询处理（多处理器）

If docs are partitioned across many machines:

- each processor carries out the search (on different doc subset)
- matching documents sent to a coordinating machine
- coordinating machine performs merging and presentation of matches

如果文档在许多计算机上进行分区：

每个处理器执行搜索（在不同的doc子集上）

匹配发送到协调机器的文档

协调机器执行匹配的合并和表示

Result Ranking
结果排名

Use query independent score 使用查询独立分数

Use query dependent score 使用查询依赖分数

How to improve efficiency? 如何提高效率？

Improving Efficiency
提高效率

Order postings list by importance of term in doc:
- as the search continues, docs become less likely

Order postings list by importance of doc itself:
- more important web documents are earlier on list

Cache (save) the results of popular queries - no search needed

按文档中术语的重要性排序过帐列表：随着搜索的继续，文档变得不太可能

根据doc本身的重要性排序发布列表：更重要的Web文档在列表中较早

缓存（保存）热门查询的结果 - 无需搜索

Summary of Web Search

Based on:
Inverted Indexing
TF*IDF
Other measures (font size, position in doc etc)
PageRank
Click data

Uses:
Parallelism
Multiple standard PCs
Continual crawling

Must continually become faster to stay the same speed!