



Search Engines--Information Retrieval in Practice

tanshuqiu

Email: tsq@cqut.edu.cn

Crawls and Feeds

• First : Deciding What to Search •

• Second : Crawling the Web •

• Third : Crawling Documents and Email •

• Fourth : Document Feeds •

Crawls and Feeds

Fifth : The Conversion Problem

Sixth : Storing the Documents

Seventh : Detecting Duplicates

Eighth : Removing Noise

Deciding What to Search

“What should we search?” The simple answer is everything you possibly can.

Crawling the Web

“Finding and downloading web pages automatically is called **crawling**, and a program that downloads pages is called a **web crawler**.

Most organizations do not have enough storage space to store even a large fraction of the Web, but web search providers with plenty of resources must still constantly download new content to keep their collections current.

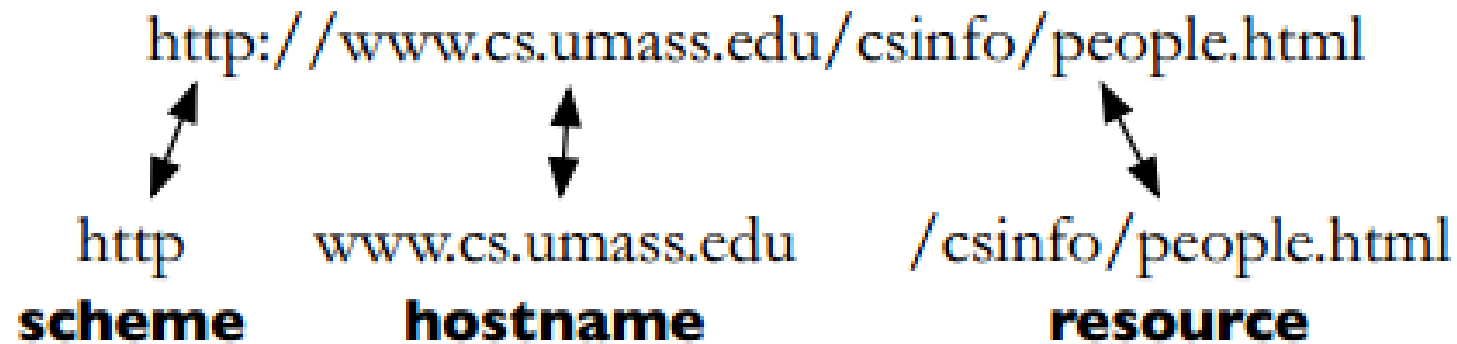
Crawling the Web

Contents

- ✓ Retrieving Web Pages
- ✓ The Web Crawler
- ✓ Freshness
- ✓ Focused Crawling
- ✓ Deep Web
- ✓ Sitemaps

Retrieving Web Pages

Each web page on the Internet has its own unique uniform resource locator, or **URL**. Any URL used to describe a web page has three parts: the scheme, the hostname, and the resource name.

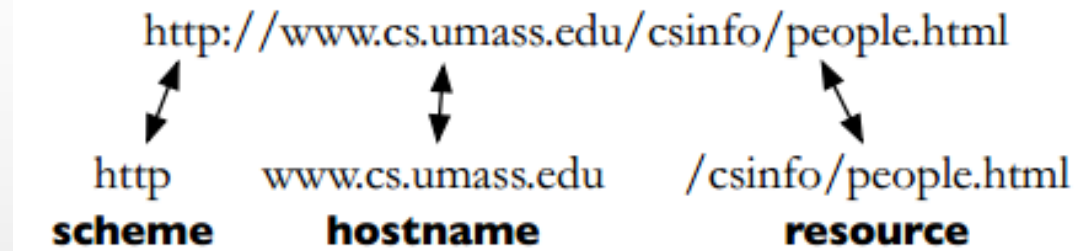


The diagram illustrates the components of a URL. At the top, the full URL `http://www.cs.umass.edu/csinfo/people.html` is shown. Below it, three components are identified with double-headed arrows pointing to their respective parts in the URL above: `http` is labeled **scheme**, `www.cs.umass.edu` is labeled **hostname**, and `/csinfo/people.html` is labeled **resource**.

```
http://www.cs.umass.edu/csinfo/people.html
```

`http` **scheme** `www.cs.umass.edu` **hostname** `/csinfo/people.html` **resource**

Retrieving Web Pages



Web pages are stored on web servers, which use a protocol called Hypertext Transfer Protocol, or HTTP, to exchange information with client software.

Hostname is the name of the computer that is running the web server that holds this web page.

Retrieving Web Pages

Web browsers and web crawlers are two different kinds of web clients, but both fetch web pages in the same way.

First, the client program connects to a domain name system (DNS) server.

Once the connection is established, the client program sends an HTTP request to the web server to request a page. The most common HTTP request type is a GET request.

Retrieving Web Pages

This simple request asks the server to send the page called `/csinfo/people.html` back to the client, using version 1.0 of the HTTP protocol specification.

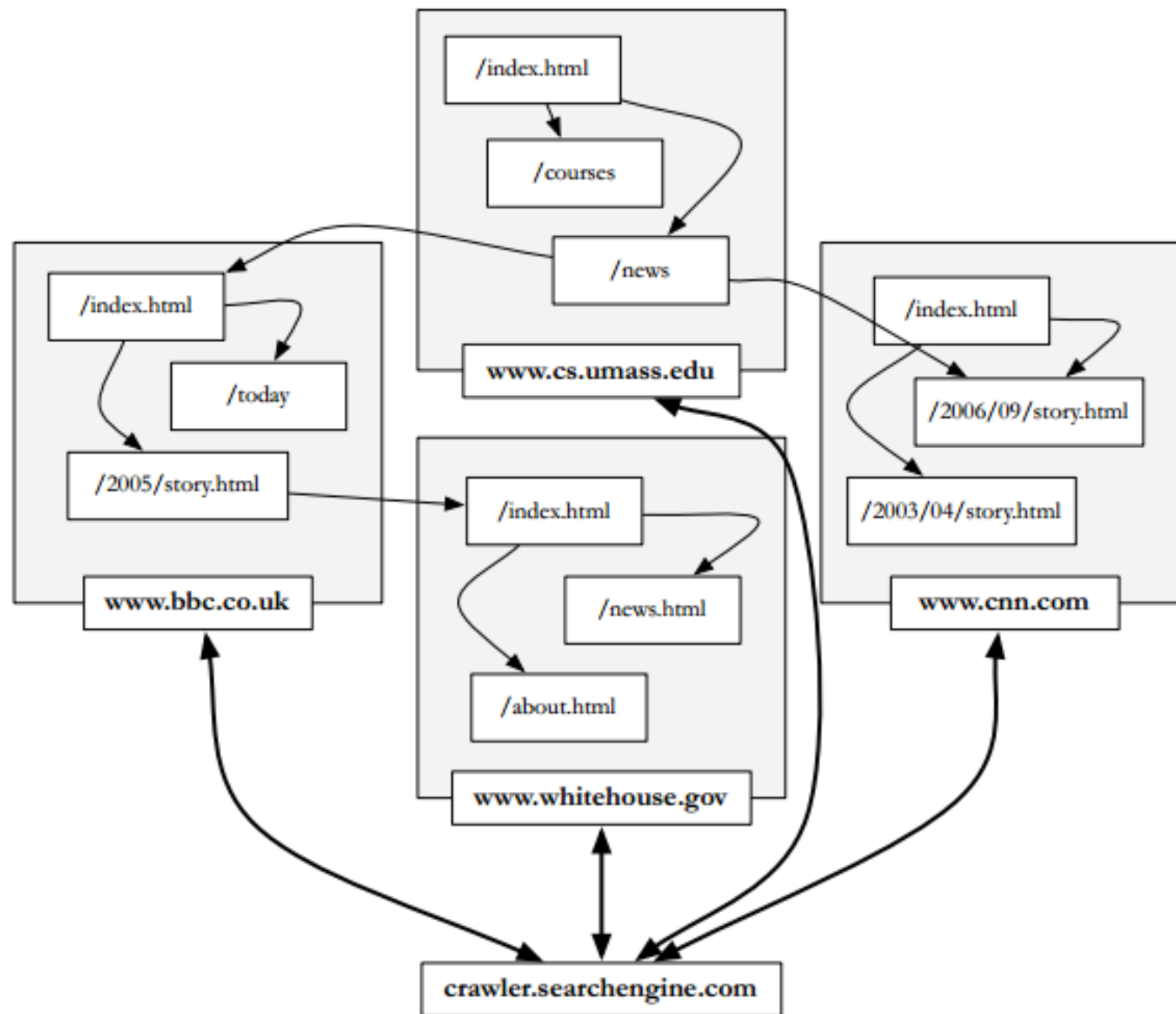
```
GET /csinfo/people.html HTTP/1.0
```

Retrieving Web Pages

A client can also fetch web pages using POST requests. A POST request is like a GET request, except that it can send additional request information to the server. By convention, GET requests are used for retrieving data that already exists on the server, whereas POST requests are used to tell the server something. A POST request might be used when you click a button to purchase something or to edit a web page.

The Web Crawler

The web crawler has two jobs: downloading pages and finding URLs.



The Web Crawler

The crawler starts with a set of seeds, which are a set of URLs given to it as parameters. These seeds are added to a URL request queue. The crawler starts fetching pages from the request queue. Once a page is downloaded, it is parsed to find link tags that might contain other useful URLs to fetch.

The Web Crawler

If the crawler finds a new URL that it has not seen before, it is added to the crawler's request queue, or frontier. The frontier may be a standard queue, or it may be ordered so that important pages move to the front of the list. This process continues until the crawler either runs out of disk space to store pages or runs out of useful links to add to the request queue.

The Web Crawler

If the crawler finds a new URL that it has not seen before, it is added to the crawler's request queue, or frontier. The frontier may be a standard queue, or it may be ordered so that important pages move to the front of the list. This process continues until the crawler either runs out of disk space to store pages or runs out of useful links to add to the request queue.

Freshness

Web pages are constantly being added, deleted, and modified. To keep an accurate view of the Web, a web crawler must continually revisit pages it has already crawled to see if they have changed in order to maintain the freshness of the document collection.

Focused Crawling

Some users would like a search engine that focuses on a specific topic of information. For instance, at a website about movies, users might want access to a search engine that leads to more information about movies. If built correctly, this type of vertical search can provide higher accuracy than general search because of the lack of extraneous information in the document collection.

Focused crawlers require some automatic means for determining whether a page is about a particular topic.

Deep Web

Not all parts of the Web are easy for a crawler to navigate. Sites that are difficult for a crawler to find are collectively referred to as the **deep Web** (also called the hidden Web).

Deep Web

Most sites that are a part of the deep Web fall into three broad categories:

Private sites.

Form results.

Scripted pages.

Deep Web

Private sites are intentionally private. They may have no incoming links, or may require you to log in with a valid account before using the rest of the site. These sites generally want to block access from crawlers.

Deep Web

Form results are sites that can be reached only after entering some data into a form. For example, websites selling airline tickets typically ask for trip information on the site's entry page. You are shown flight information only after submitting this trip information.

Deep Web

Scripted pages are pages that use JavaScript™, Flash®, or another client-side language in the web page. If a link is not in the raw HTML source of the web page, but is instead generated by JavaScript code running on the browser, the crawler will need to execute the JavaScript on the page in order to find the link. Although this is technically possible, executing JavaScript can slow down the crawler significantly and adds complexity to the system.

Distributed Crawling

For crawling individual websites, a single computer is sufficient. However, crawling the entire Web requires many computers devoted to crawling.

Distributed Crawling

One reason to use multiple computers is to put the crawler closer to the sites it crawls. Long-distance network connections tend to have lower throughput(fewer bytes copied per second) and higher latency (bytes take longer to cross the network).

Distributed Crawling

Another reason for multiple crawling computers is to reduce the number of sites the crawler has to remember. A crawler has to remember all of the URLs it has already crawled, and all of the URLs that it has queued to crawl.

Yet another reason is that crawling can use a lot of computing resources, including CPU resources for parsing and network bandwidth for crawling pages.

Crawling Documents and Email

“Since websites are meant to be viewed with web browsers, most web content is stored in HTML. On the other hand, each desktop program—the word processor, presentation tool, email program, etc.—has its own file format. So, just finding these files is not enough; eventually they will need to be converted into a format that the indexer can understand.

Document Feeds

A document feed is particularly interesting for crawlers, since the crawler can easily find all the new documents by examining only the end of the feed.

We can distinguish two kinds of document feeds, **push** and **pull**.

Document Feeds

A **push** feed alerts the subscriber to new documents. This is like a telephone, which alerts you to an incoming phone call; you don't need to continually check the phone to see if someone is calling.

A **pull** feed requires the subscriber to check periodically for new documents; this is like checking your mailbox for new mail to arrive.

The Conversion Problem

Standard text file formats include raw text, RTF, HTML, XML, Microsoft Word, ODF (Open Document Format) and PDF (Portable Document Format). There are tens of other less common word processors with their own file formats.

The most common way to handle a new file format is to use a conversion tool that converts the document content into a tagged text format such as HTML or XML.

Character Encodings

The text that you see on this page is a series of little pictures we call letters or glyphs. Of course, a computer file is a stream of bits, not a collection of pictures. A character encoding is a mapping between bits and glyphs.

Character Encodings

To solve this mess of encoding issues, Unicode was developed. Unicode is a single mapping from numbers to glyphs that attempts to include all glyphs in common use in all known languages. This solves the problem of using multiple languages in a single file. Unfortunately, it does not fully solve the problems of binary encodings, because Unicode is a mapping between numbers and glyphs, not bits and glyphs. It turns out that there are many ways to translate Unicode numbers to glyphs!

Storing the Documents

After documents have been converted to some common format, they need to be stored in preparation for indexing.

Most other kinds of search engines need to store documents somewhere. Fast access to the document text is required in order to build document snippets² for each search result. These snippets of text give the user an idea of what is inside the retrieved document without actually needing to click on a link.

Storing the Documents

Using a Database System

If you have used a relational database before, you might be thinking that a database would be a good place to store document data. For many applications, in fact, a database is an excellent place to store documents. A database takes care of the difficult details of storing small pieces of data, such as web pages, and makes it easy to update them later.

Storing the Documents

Random Access

To retrieve documents quickly in order to compute a snippet for a search result, the document store needs to support random access. Compared to a full relational database, however, only a relatively simple lookup criterion is needed. We want a data store such that we can request the content of a document based on its URL.

Storing the Documents

Compression and Large Files

Compression works best with large blocks of data, which makes it a good fit for big files with many documents in them. Most compression methods do not allow random access, so each block can only be decompressed sequentially. If you want random access to the data, it is better to consider compressing in smaller blocks, perhaps one block per document, or one block for a few documents. Small blocks reduce compression ratios but improve request latency.

Storing the Documents

Update

As new versions of documents come in from the crawler, it makes sense to update the document store. The alternative is to create an entirely new document store by merging the new, changed documents from the crawler with document data from the old document store for documents that did not change.

Detecting Duplicates

Documents with very similar content generally provide little or no new information to the user, but consume significant resources during crawling, indexing, and search. In response to this problem, algorithms for detecting duplicate documents have been developed so that they can be removed or treated as a group during indexing and ranking.

Detecting Duplicates

Detecting exact duplicates is a relatively simple task that can be done using check summing techniques. A checksum is a value that is computed based on the content of the document. The most straightforward checksum is a sum of the bytes in the document file.

Detecting Duplicates

For example, the checksum for a file containing the text “Tropical fish” would be computed as follows (in hex):

Any document file containing the same text would have the same checksum.

T	r	o	p	i	c	a	l		f	i	s	h	<i>Sum</i>
54	72	6F	70	69	63	61	6C	20	66	69	73	68	508

Detecting Duplicates

The detection of near-duplicate documents is more difficult. Even defining a near-duplicate is challenging. Web pages, for example, could have the same text content but differ in the advertisements, dates, or formatting. Other pages could have small differences in their content from revisions or updates. In general, a near-duplicate is defined using a threshold value for some similarity measure between pairs of documents.

Removing Noise

Many web pages contain text, links, and pictures that are not directly related to the main content of the page.

From the perspective of the search engine, this additional material in the web page is mostly noise that could negatively affect the ranking of the page.

Removing Noise

A major component of the representation of a page used in a search engine is based on word counts, and the presence of a large number of words unrelated to the main topic can be a problem. For this reason, techniques have been developed to detect the content blocks in a web page and either ignore the other material or reduce its importance in the indexing process.