*

Finding Needles in the World's Biggest Haystack

- or -

The Page Match Problem

ISC1057

Janet Peterson and John Burkardt

Computational Thinking

Fall Semester 2016

In this discussion, we will look at how a search engine deals with an impossible problem:

- Given a word or phrase that you just typed in, search the 4.6 billion pages on the Internet and list all the pages that contain that phrase
- AND...
- Do this in two seconds or less.

To most of us, the internet is simply an enormous pile of

- documents;
- pictures;
- messages;
- songs;
- magazines;
- movies;
- programs;
- classified ads;
- retail stores and ticket outlets.


STORAGE BOXES
Cottonelle

The Internet may seem to be able to connect us to everything; but if we don't know where something is, we might as well not have it!

So as the Internet has grown in size (number of things connected) and complexity (kinds of things connected), it has become increasingly important to develop and improve ways of:

- finding things you are looking for;
- recommending things you might not know about, but might be interested in;
- learning enough about your interests to guess what other things you might want to see.


NETFLIX


goodreads


Expedia


mapquest


yelp

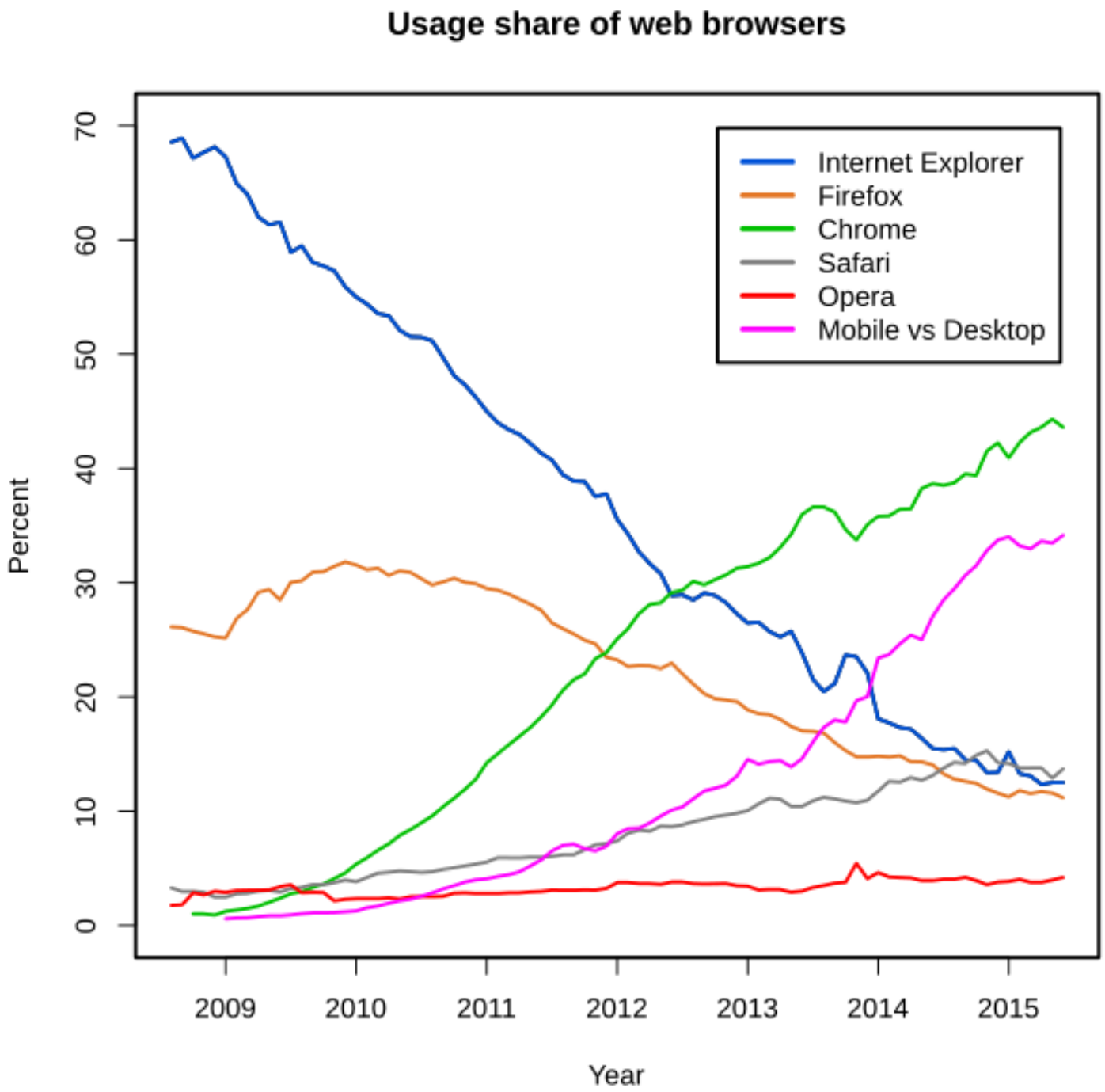

YouTube

Examples of these aides to searching and using the Internet include NetFlix, GoodReads, MapQuest, Expedia, Yelp, YouTube.

We think of these as convenient guides to specialized information.

After entering some search words, we expect to download a desired song, video or movie, to make a reservation for a meal, a plane trip, a music concert, or a sports event, or to determine a specific far away location, and the best route for reaching it.

A program like Yelp might seem to have a simple task: handle certain simple requests *(where is the nearest Thai restaurant?)* or actions *(I wish to report that the food here is not vegan!)*.


Usage share of web browsers
Internet Explorer
Firefox
Chrome
Safari
Opera
Mobile vs Desktop
Percent
0
10
20
30
40
50
60
70
2009
2010
2011
2012
2013
2014
2015
Year

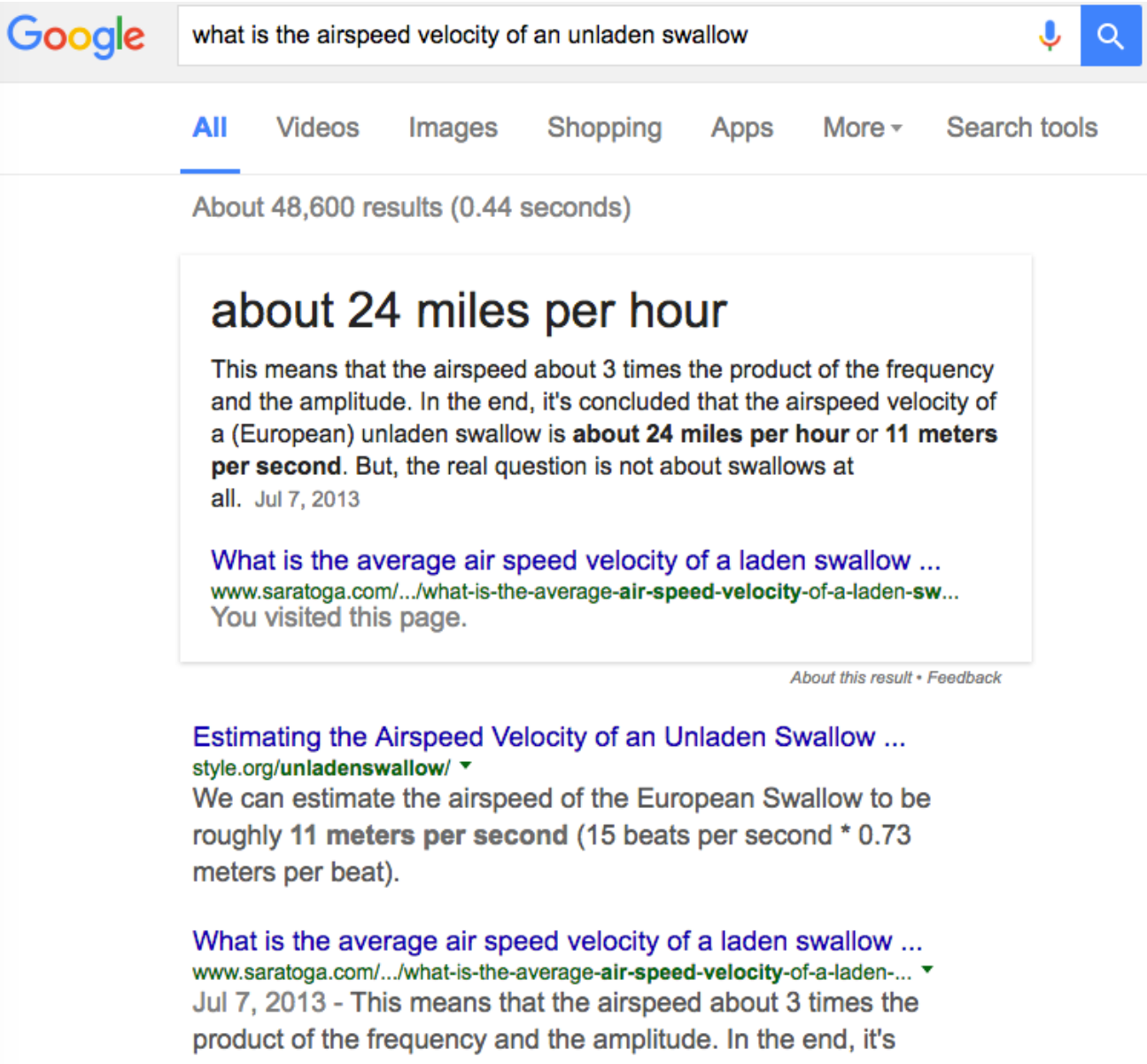
But suppose you are looking for a list of the rulers of Russia in the 18th century, or advice on how to whistle, or instructions on how to get rid of a wasp infestation.

One of the most useful features of a browser is its connection to a search engine, which will hunt for information on the Internet, guided by a word, or several words, or quoted words in a specific order.

So what actually happens when you type something like:

"What is the air speed velocity of an unladen swallow?"

into Chrome, FireFox, Opera, Safari or good old Windows Explorer?


Google
what is the airspeed velocity of an unladen swallow
All
Videos
Images
Shopping
Apps
More
Search tools
About 48,600 results (0.44 seconds)
about 24 miles per hour
This means that the airspeed about 3 times the product of the frequency and the amplitude. In the end, it's concluded that the airspeed velocity of a (European) unladen swallow is about 24 miles per hour or 11 meters per second. But, the real question is not about swallows at all. Jul 7, 2013
What is the average air speed velocity of a laden swallow ...
www.saratoga.com/.../what-is-the-average-air-speed-velocity-of-a-laden-sw...
You visited this page.
About this result • Feedback
Estimating the Airspeed Velocity of an Unladen Swallow ...
style.org/unladenswallow/
We can estimate the airspeed of the European Swallow to be roughly 11 meters per second (15 beats per second * 0.73 meters per beat).
What is the average air speed velocity of a laden swallow ...
www.saratoga.com/.../what-is-the-average-air-speed-velocity-of-a-laden-...
Jul 7, 2013 - This means that the airspeed about 3 times the product of the frequency and the amplitude. In the end, it's

Your browser is running on the computer right in front of you.

The information you need is somewhere else, perhaps far away.

We assume you've got a connection, wireless or wired, to your local network. Your request goes out from the local network to the Internet... and a few seconds later, your answer appears, a list of hundreds or thousands of "hits", showing about 15 or 20 on the first page, including the location and an extract from the matching text.

It seems like magic, and in fact, it is impossible for a search engine to receive a search string and check every word of every web page on the whole Internet and return a list of the good matches to you unless you are willing to wait **days** for an answer.

But we got the answer in two seconds!

What makes the impossible possible?

Why do we say the job is **impossible**?

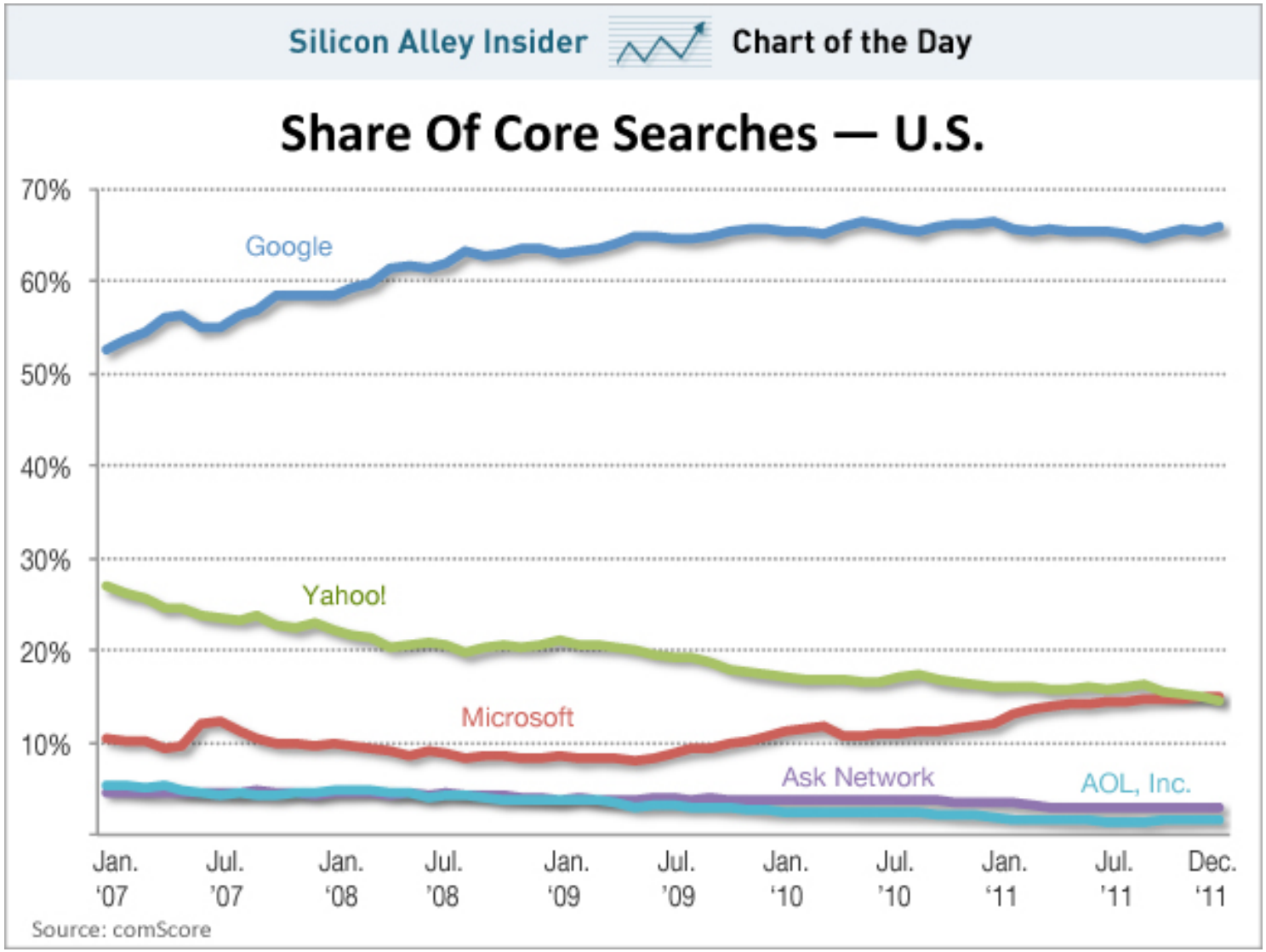
To access a single (small!) web page:

- the browser must convert the name of the web address to a numeric IP (Internet Protocol) address;
- setup a connection to that address, request a copy of the page;
- wait for the remote server to respond;
- wait for the web page to be delivered.

Each single web page access can take on the order of 1 second.

There are more than 4 billion web pages on the Internet;

To copy every web page would take 4 billion seconds = 124 years.


Silicon Alley Insider
Chart of the Day
Share Of Core Searches — U.S.
70%
60%
50%
40%
30%
20%
10%
Google
Yahoo!
Microsoft
Ask Network
AOL, Inc.
Jan. '07
Jul. '07
Jan. '08
Jul. '08
Jan. '09
Jul. '09
Jan. '10
Jul. '10
Jan. '11
Jul. '11
Dec. '11
Source: comScore

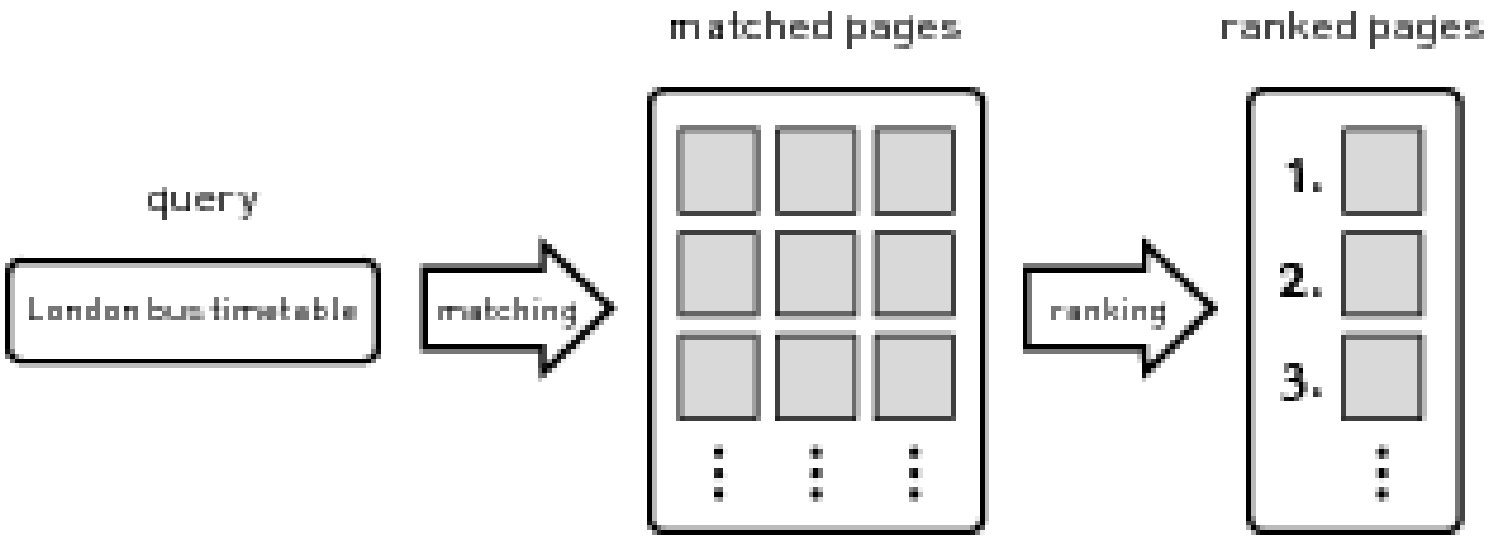
We are assuming that when we type a search string, the browser really goes out and checks every web page, right then, to get our result.

But since that is impossible, and we nonetheless get our results, then something very clever must be going on.

The solution to the seemingly impossible task of finding pages that match a search string is known as search engine indexing.

A better implementation of search engine indexing is a crucial weapon in the "war" between search engine companies.

In 2002, Google, Yahoo, and MSN had about 30% of the search engine market share, but Google made some dramatic improvements to its search engine, and drove Yahoo and MSN down to less than 20% shares each (and dropping).



When you issue a web search query, it is processed in two stages:

- matching searches for all matching pages;
- ranking orders the matching pages so the best appear first.

Thus, if our search string is "London bus timetable", then the matching step finds all web pages that match this string, and the ranking step arranges the matches in order.

It is possible to consider these steps independently, and so for now, we will concentrate only on the matching problem.

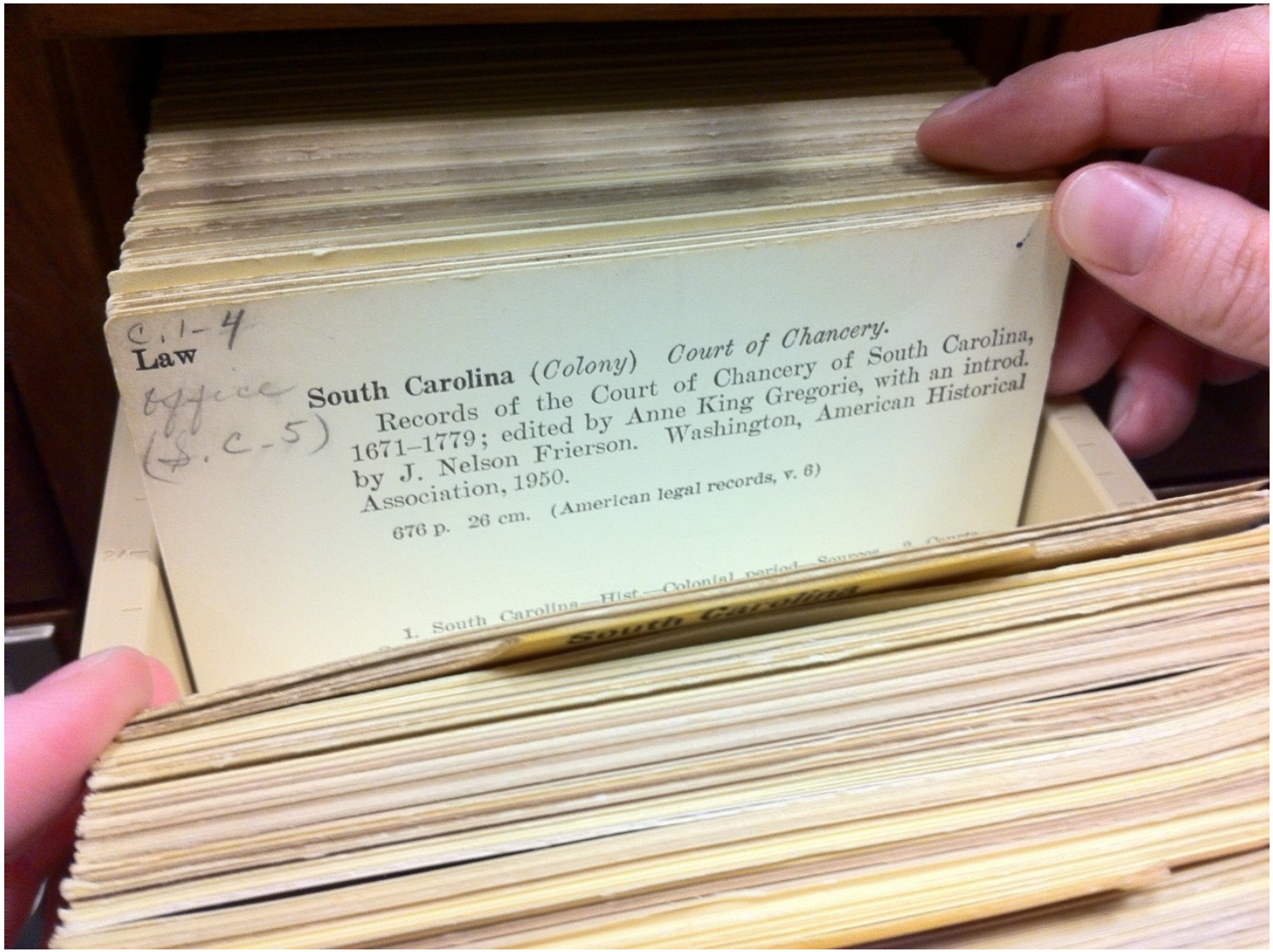
One way to realize how search engine indexing makes the page match task possible is to think about what used to happen, in the good old days, when you went to the library to work on a term paper.

You could go to a librarian and ask

"What is the air speed velocity of an unladen swallow?"

Did the librarian then run into the stacks of books and take down the books, one at a time, scanning each one for the search phrase?

The librarian said: Look in the catalog card index!


C.1-4
Law
Office
(S.C-5)
South Carolina (Colony) Court of Chancery.
Records of the Court of Chancery of South Carolina, 1671–1779; edited by Anne King Gregorie, with an introd. by J. Nelson Frierson. Washington, American Historical Association, 1950.
676 p. 26 cm. (American legal records, v. 6)
1. South Carolina—Hist.—Colonial period—Sources. 2. Courts
South Carolina

The old card catalog index contained hundreds of thousands of search phrases: topics, author names, events, all in alphabetical order.

Every time a new book came into the library, the librarians prepared cards for every topic covered by the book, and added these to the index.

With great work, and over a long time, the card catalog was built up to be a labor saving device that allowed you to "instantly" (well, within a minute or two) discover the "addresses" (Library call numbers) of every book in the library that might pertain to your topic.

As another example, most nonfiction books include an index, which lists names and topics in alphabetical order, and the pages on which these are discussed.

For instance, every time a politician in Washington D.C. publishes a book, everyone rushes to the bookstore, goes to the back of the book, and looks to see if their own name appears in the index!

An index takes a long time to make; most of the information will never actually be used, and it's important to select only useful occurrences of important items.

But once the index is prepared, then it's possible to rapidly locate any indexed item, no matter how big the book.

Thus, the first ingredient in our rapid page matching procedure is something we can call **The Indexing Trick**.

To see how indexing might solve our page matching problem, let's start with a simple model of the world of web pages, in which there are just three short pages to consider, which we will call pages 1, 2 and 3.

Indexing these web pages is very similar to indexing a book. We will consider every word important. So we start by making an alphabetical list of all the words that occur. Then each word will be followed by a "1" if it occurs in page 1, a "2" if in page 2, and so on.

Let's go through this painful indexing process now.

Our resulting index should look like this. It is a single file that contains all the words that appear, and which web pages use them:

a	3
cat	1 3
dog	2 3
mat	1 2
on	1 2
sat	1 3
stood	2 3
the	1 2 3
while	3

If a search engine had just this single index file, it could already answer a number of questions.

If we enter the search string $\mathbf{dog}$, the search engine can quickly find the corresponding line in the index (this is quick, because a computer takes advantage of alphabetical ordering even more efficiently than we do). Then the search engine can immediately report that "dog" appears in pages 2 and 3. Of course, a real search engine would include a bit of the text surrounding the occurrence of "dog" but that's an added feature that we don't need to worry about right now.

If we enter the search string $\mathbf{cat}$, our search engine will tell us that this word occurs in pages 1 and 3.

And if we enter the search string $\mathbf{dog\ cat}$ then the search engine can first determine that "dog" occurs in pages 2 and 3; it then looks up "cat", but notices that although that string occurs on pages 1 and 3, only page 3 has both words, and so the correct response is page 3.

Notice an important fact. In order to answer these questions, we did not have to have access to the original web pages. We needed that when we made the index, but now a single index file allows us to answer questions about all three pages.

Suppose, then, that we had 4 billion web pages, and we were somehow able to create a similar index file for them. Then, in order to answer these simple matching questions about all the web pages, we only have to search one file, and we don't need any access to the Internet.

This is one clue to how a task that should take 124 years can be cut down to 2 seconds.

In most search engines, it is possible to enter a phrase, using quotation marks, such as "cat sat". In that case, you are not just asking that both words appear on a given page, but that they appear immediately together, in that order.

Our first index file can tell us that both $\mathbf{cat}$ and $\mathbf{sat}$ occur on pages 1 and 3. But this does not tell us these two words occur together.

It might seem that the solution might be to look up $\mathbf{cat}$ and then go to those web pages and find whether $\mathbf{sat}$ occurs in the right position.

This is not acceptable! It still requires access to an unknown number of web pages, and searching those web pages for every occurrence of $\mathbf{cat}$, which means we cannot return a response in time.

Now suppose that we make a second version of our index, but this time:

- we label every word in each page with its position;
- we record every occurrence of a word in the page along with its position.

a	3-5				
cat	1-2	3-2			
dog	2-2	3-6			
mat	1-6	2-6			
on	1-4	2-4			
sat	1-3	3-7			
stood	2-3	3-3			
the	1-1	1-5	2-1	2-5	3-1
while	3-4				

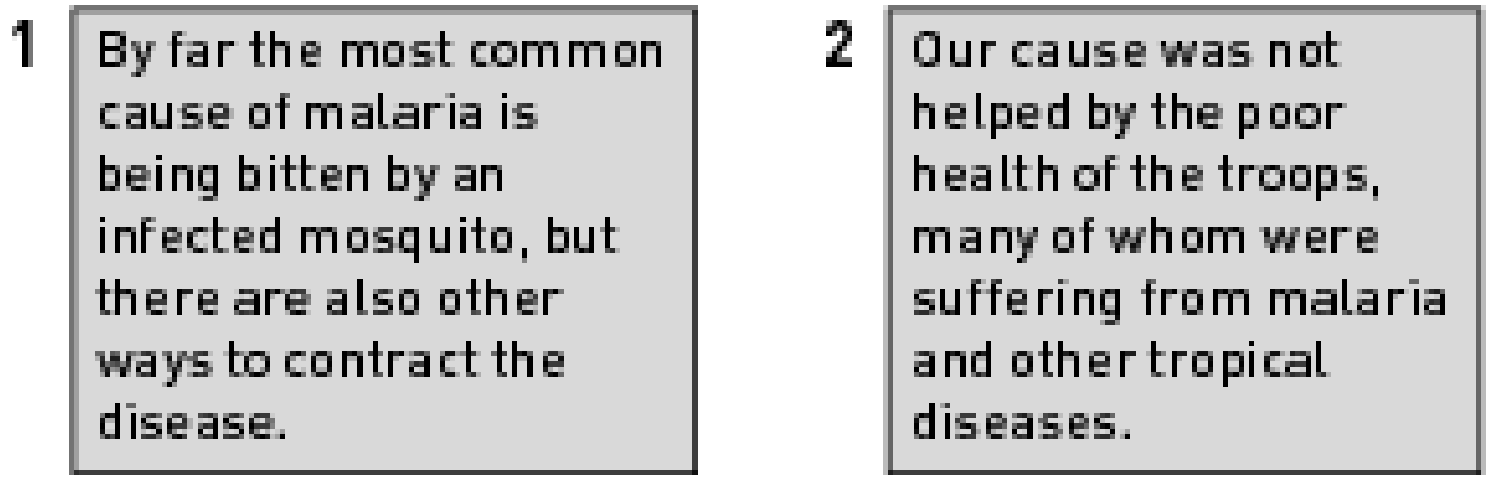
Now suppose we are given the search phrase "cat sat".

We look up the word **cat**, and see that it occurs on page 1 as word 2, and on page 3 as word 2.

Now we look up the word **sat** and see that it occurs on page 1 as word 3, right after **cat**, so we have a hit on page 1. It also occurs on page 3, but as word 7, so it does not immediately follow **cat** there, and so that counts as a miss.

By making our index more intelligent, we can now answer *any* phrase inquiry about our web pages, without needing to access the original pages.

We can call this **The Word Location Trick**.



Suppose we were interested in learning the cause of malaria. We might naturally search on **malaria cause** although we probably don't insist that those two words occur exactly together.

Suppose the search engine discovered two web pages with both match words. We can see the first web page is a better match. What clues could a search engine use?

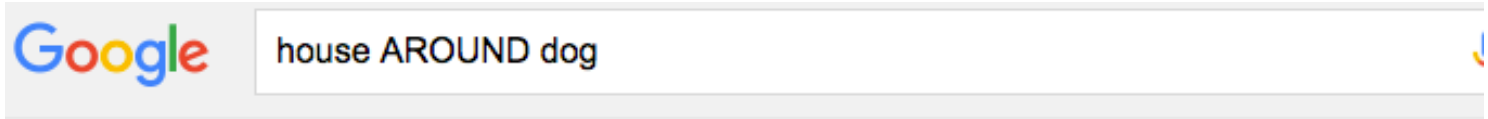
In page 1, the two search words are close together, while in page 2

they are not. This suggests that page 1 is a better match.

by	1-1			
cause	1-6	2-2		
common	1-5			
...				
malaria	1-8	2-19		
many	2-13			
of	2-10	2-14		
...				
the	1-3	1-24	2-7	2-11

With a word location index, the search engine can see that on page 1, **malaria** and **cause** are just 2 apart, versus 17 apart on page 2, suggesting page 1 is the better match.

The **Nearness Trick** is useful as part of the page ranking process.



About 411,000,000 results (0.56 seconds)

Petco® - Dog Houses - Shop Dog Houses & More
Ad www.petco.com/Dog
4.2 ★★★★☆ rating for petco.com
Free Shipping on Most Orders.
Brands: You & Me, Petco, Pet Gear, Snoozer, Pet Ego
Shop with Confidence – Google Trusted Stores

Why Does My Dog Race Through My House? - Vetstreet
www.vetstreet.com › Dr. Marty Becker
Aug 9, 2012 - Q. Why does a perfectly relaxed **dog** jump up and start tearing **around** the **house** at a hundred miles an hour out of nowhere? A. Trainers and ...

Why Do Dogs Zoom-zoom-zoom Around the House ...
https://natural**dog**training.com/.../why-do-**dogs**-zoom-zoom-zoom-aroun...
Jun 1, 2009 - Behaviorists call the syndrome of a **dog** running helter-skelter **around** the yard, or zooming from room to room in the **house** "frequent, random ...

We already know two ways to specify search words:

- quoted, the words must appear together, in that order;
- unquoted, the words can be in any order and far apart.

However, in some search engines, it is possible to request that the search words simply be near each other. This is sometimes called a **proximity search**.

In Google Search, for instance, we can ask for pages that include **house** and **dog** near each other using the words:

```
house AROUND Dog
```

One reason that search engines also prefer matches in which multiple keywords are close is to avoid being trapped by spamdexes. A spamdex is an artificial web page that simply contains a grab bag of keywords, without any information. You could make such a web page by posting a dictionary, minus the definitions, for instance.

A search engine looking for **hair loss remedy** or **tap dance lessons** or **perpetual motion machines** will find matches (but no information!) on a spamdex page, and the spamdex operator will pick up some money by displaying ads to the annoyed user.

Thus, even if the user doesn't request that the keywords be close, search engines avoid matching pages that fail the proximity test.

Web pages are actually a little more complicated than the simple text files we have used so far as examples.

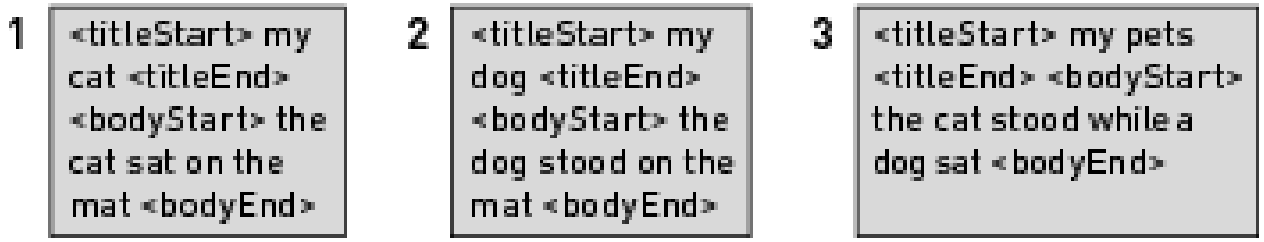
Web pages are written in a special language called **HTML**, the HyperText Markup Language. HTML allows the author to vary the font type and size, to include tables, lists and figures, and to indicate the structure of the document.

In particular, a web page author can specify a title for the web page.

If a search engine finds a web page with the word **malaria**, doesn't it make a huge difference if the actual title of the page is "Malaria"?

Here, we see three web pages with titles.

The pages as we see them.

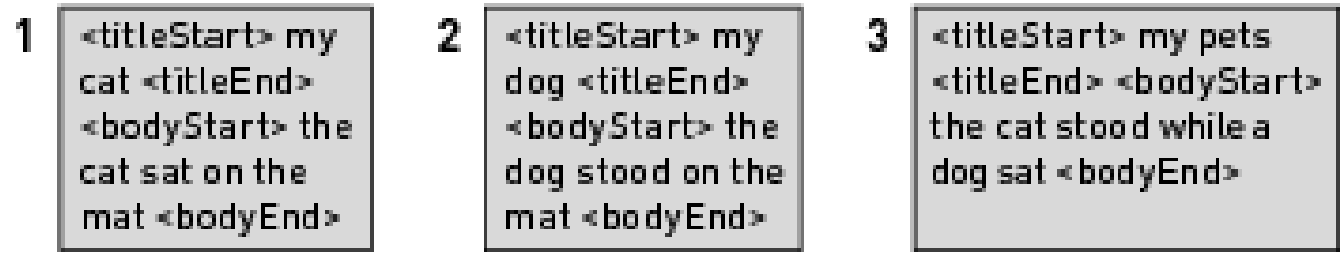


The pages as the browser and search engine see them.

A web page usually has a **title** between special markers (metawords) **<titleStart>** and **<titleEnd>**.

An intelligent search engine takes advantage of noticing the title!

(Note that the actual markers used by HTML are slightly different.)



Our third index attempt includes HTML key words.

a	3-10				
cat	1-3	1-7	3-7		
dog	2-3	2-7	3-11		
mat	1-11	2-11			
my	1-2	2-2	3-2		
on	1-9	2-9			
pets	3-3				
sat	1-8	3-12			
stood	2-8	3-8			
the	1-6	1-10	2-6	2-10	3-6
while	3-9				
<bodyEnd>	1-12	2-12	3-13		
<bodyStart>	1-5	2-5	3-5		
<titleEnd>	1-4	2-4	3-4		
<titleStart>	1-1	2-1	3-1		

Suppose a user searches for **dog**. A page in which **dog** is in the title is probably a stronger match.

Each time a page is found containing the word **dog**, the engine can check whether this word is actually part of the web page title. It does this by comparing the positions of **<titleEnd>** and **dog** and **<titleStart>**. If the keyword falls between the two title markers, then this web page is more highly related than if it occurs elsewhere.

By looking at our index, we see the following cases:

Page	titleStart	dog	titleEnd	Start < dog < End?
2	1	3	4	yes
2	1	7	4	no
3	1	11	4	no

This technique is **The Metaword Trick**.


© Google

From what we have seen, the impossible problem of quickly responding to a request to find keywords in all the webpages in the world has become the possible problem of intelligently searching a single index file.

Just as with card catalogs and an index at the back of a book, the creation of an index file for the web takes a great deal of time, and space.

Google, for instance, has created enormous collections of computer servers whose job is to collect all the information on all the web pages and create, update, and analyze the corresponding index file. This means that the index file is actually always out of date (like Google Street View) but regularly updated piece by piece.

Even with the tricks we have described, it is common for a search engine to discover hundreds of thousands of matching web pages.

The page match algorithm is only the first half of the process of responding to your web search.

Next it will be necessary to consider the page ranking algorithm, which considers all the matching pages that have been found, and sorts them in order of importance, so that even with thousands of matches, most users know their best choice is a match on one of the first few pages.