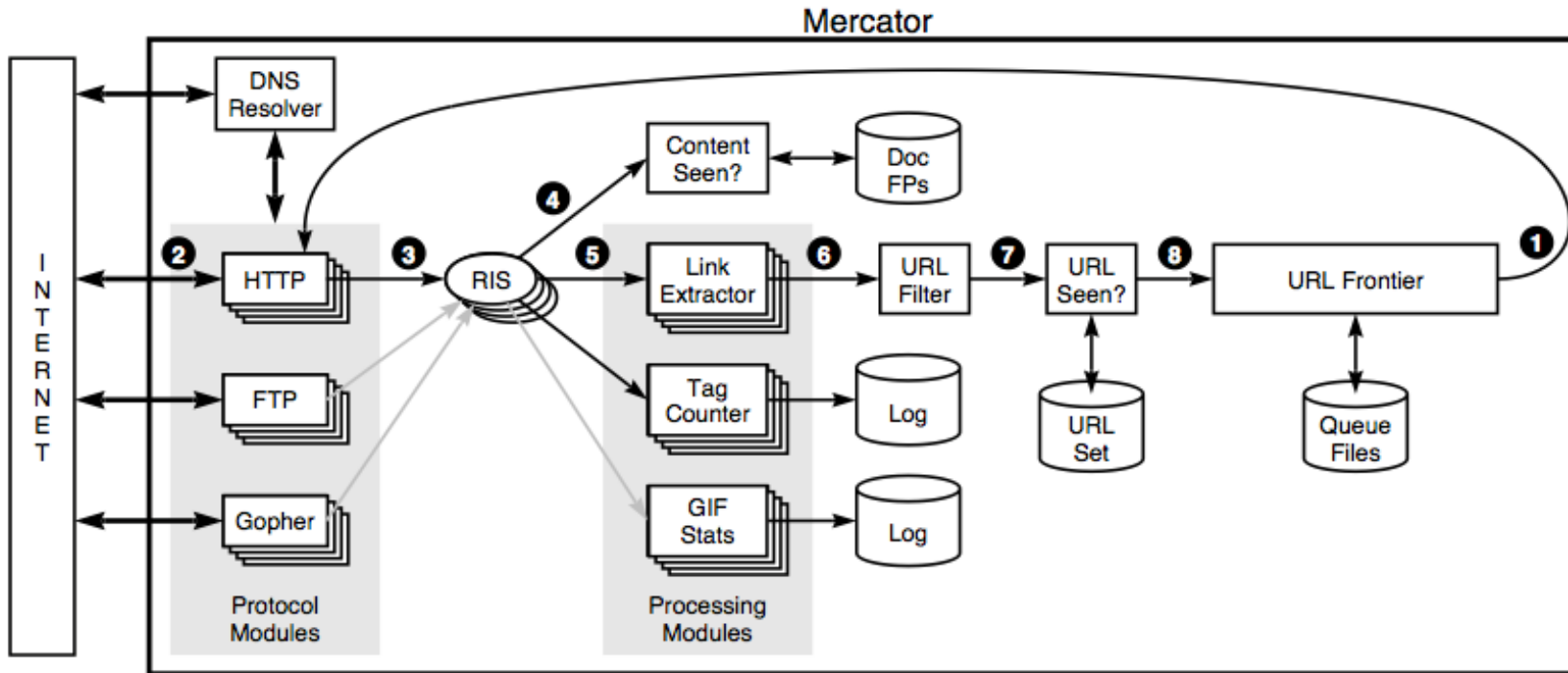


IR3: Web Search Implementation



Review: Information Retrieval 1 and 2

- Given a **query**, produce a *rank-ordered list* of **documents** from an **index** based on *relevance*
- From Part 1: Represent query and document as **tf-idf vectors**
 - Compare with **cosine similarity**
- From Part 2: Use **PageRank** to establish *influence* of each **vertex** in a **graph** of the internet

$$p_u = \frac{1-d}{N} + d \sum_{i \in e_{i,u}} \frac{p_i}{K_i}$$


- Iteratively compute p_u for each vertex. Repeat until convergence

Review: Search Engine Optimization

- **Search Engine Optimization** is a set of techniques used to **boost or alter** the order in which a **page** appears according to a search engine
 - <meta> keyword tags, alt attributes, content within document
- Expected client capabilities can influence ranking
 - Does the page require JavaScript to render?
What can your page provide if it doesn't?
 - Do you block clients with CAPTCHAs or bot detection?
- Because PageRank is affected by graph edges (i.e., links), website developers can influence a PageRank by introducing links


Link spam

- Bots add comments to blogs. Comments link back to your site.
 - Blog owner should use `rel="nofollow"` attribute for links in comments. Search engines then ignore these links.
- Search engines hate this. They "spam" filter for it.

 Written by [You](#)
1 minute ago

@Bikesgearlab, Hey thanks for your comment... It's a great example of a link scheme, because it clearly has nothing to do with my post and is entirely self serving. Good Job!

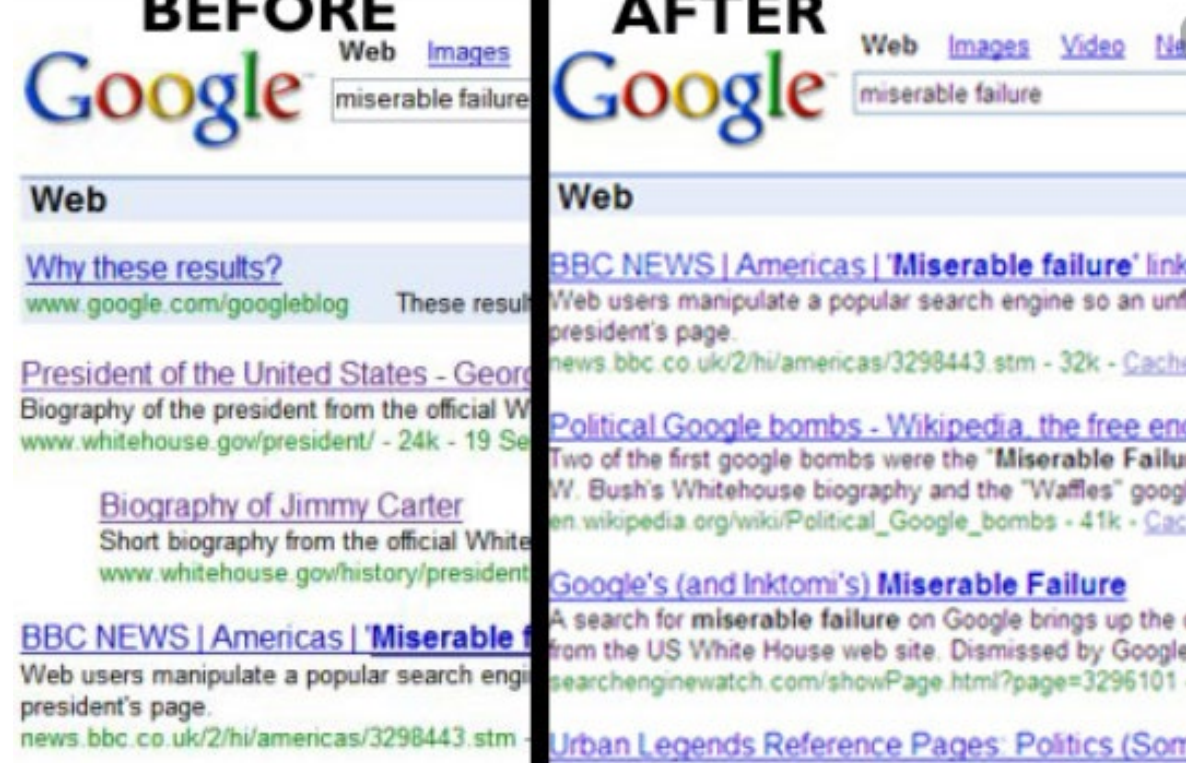
[Reply](#) | [Edit](#)

 Written by [Bikesgearlab](#)
8 minutes ago

Hey Guys,
I am here to introduce my bikes review blog.I'm try to contribute all kinds of bike reviews.I hope this blog are very very important bikes rider.Please check it just one click. <https://bikesgearlab.com>
Have A Nice Day

Google Bombing

- Recall: PageRank based on Graph structure of internet
- In practice, link text also matters
 - ` my link text `
- Search engines will consider link text when computing relevance
 - If a lot of links to URL X are associated with the same text, then words in that text must be highly relevant to the given URL
 - If a million sites have `miserable failure`, then searching for "miserable failure" is more likely to turn up "derp.com"

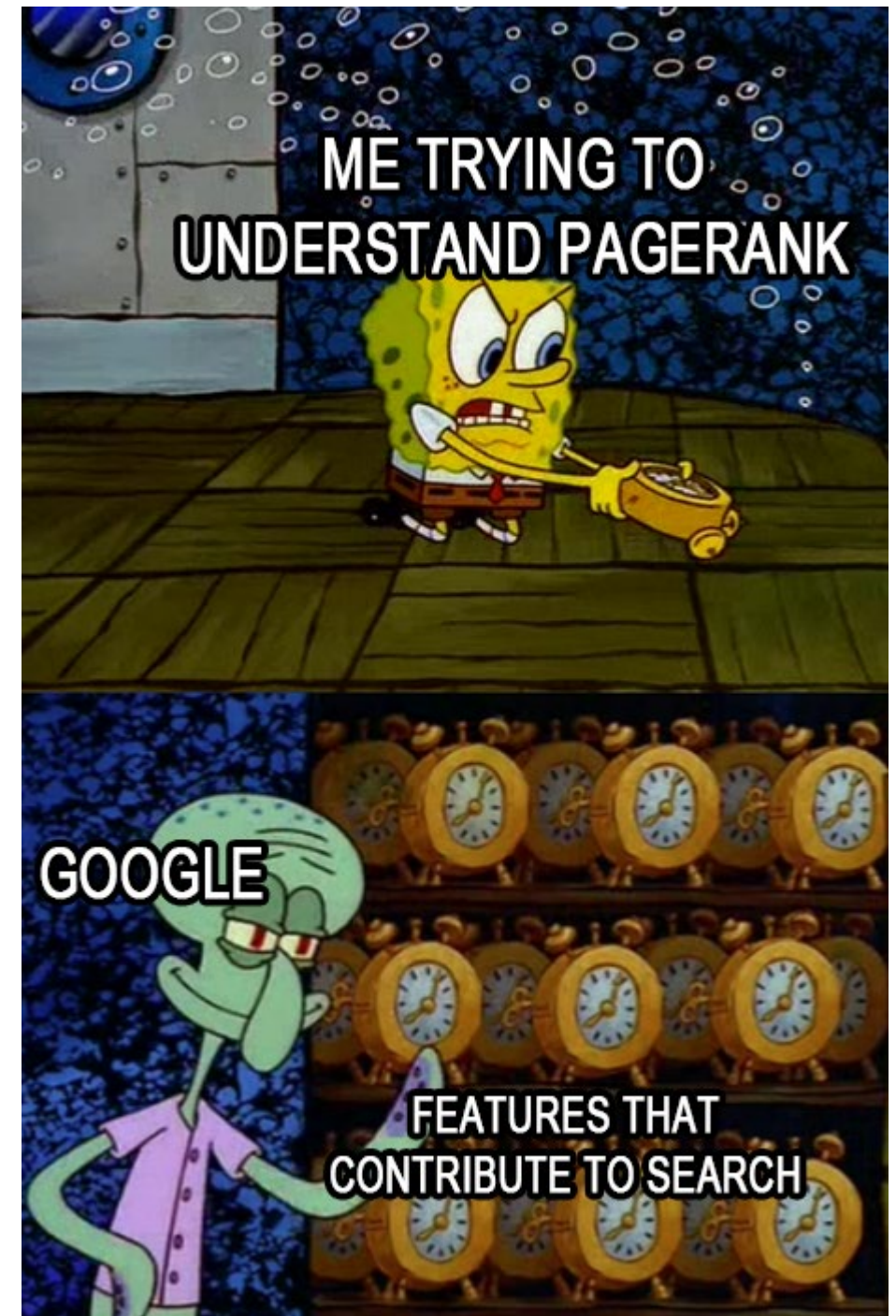


URL technical considerations

- Avoid URLs that look like form queries
 - <http://www.mysite.com/info?about>
 - <https://kjleach.eecs.umich.edu/?a=about> ☹️
- Use different URLs for different pages
 - Think about how wolverine access works. You click a link, and the URL doesn't change! Don't do this.
 - Crawlers aren't very sophisticated. They don't deal with statefulness very well
 - This should remind you of REST: have a different URL for each resource

Other relevance factors

- Google says, “Relevancy is determined by over 200 factors, one of which is the PageRank for a given page.”



One-Slide Summary: Indexing Search Engines

- We need an **index** of webpages to implement **search**
- We use **crawlers** to automatically **scrape** webpages by downloading from URLs, parsing HTML, and recursively crawling links
 - Websites can help inform crawlers about structure of a site using **robots.txt** files or special instructions about what to crawl
- We build an **inverted index** that maps *keywords* to *lists of documents*
- Search engines **deduplicate** webpages to prevent unnecessary recomputation of metadata
- **Distributed computation** can accelerate the search process

A few numbers

- 5 B - 100 B pages
 - Let's say 10 Billion for concreteness
- Assume 10KB per compressed page
 - 100 TB data to index
- 1 minute - 1 month freshness
- 3-5B queries *per day* on Google alone

Agenda

- **Crawler design**
- Inverted index construction
- Deduplication
- Distributed search architecture

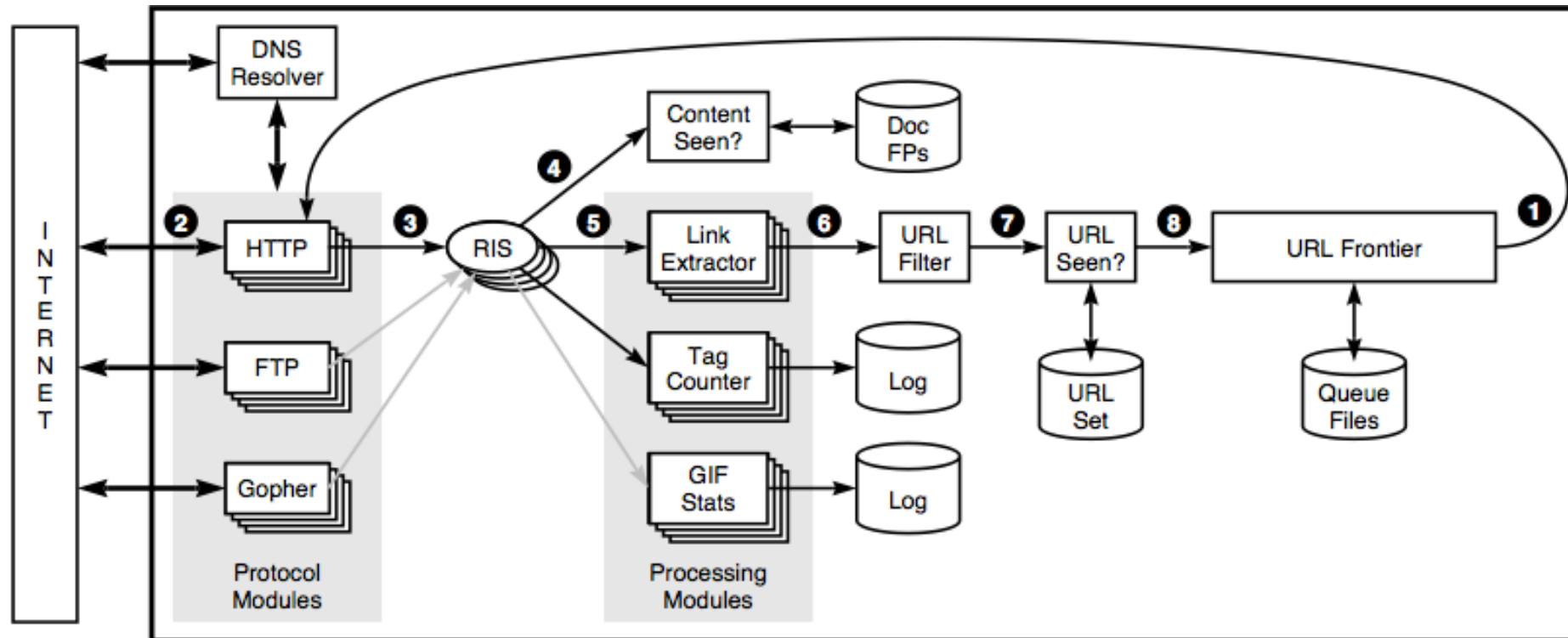
Crawlers

- To search the web, we need a program that downloads every web page
- Web Robots AKA Web Wanderers, Crawlers, or Spiders
- Example: googlebot
 - <http://www.robotstxt.org/db/googlebot.html>
- Discussion: how would you do this?
- What data structures and algorithms would you use?

Mercator

- Web crawler example: Mercator
- Mercator was the AltaVista crawler (1998)
- Exceptionally well-documented and useful to study, despite the many years
- Starts with seed URLs

Mercator



What is crawled?

- All static web pages
 - Unless restricted by `robots.txt`
- Server-side dynamic pages
 - It looks like HTML from the perspective of the Bot
 - Crawling is only as good as linking between dynamic pages
- Client-side web pages
 - GoogleBot includes a JavaScript rendering engine

Deep web

- Surface web: content indexed by search engines
- Deep web: content not indexed by search engines
- Can't index password-protected content
 - Facebook, SnapChat, Dropbox, Google products ...
 - There's a lot of it!
- Size of deep web likely several orders of magnitude larger than surface web

SURFACE WEB

Google

Bing

Wikipedia

DEEP WEB

Contains 90% of the information on the Internet, but is not accessible by Surface Web crawlers.

Academic Information

Medical Records

Legal Documents

Scientific Reports

Subscription Information

Multilingual Databases

Financial Records

Government Resources

Competitor Websites

Organization-specific
Repositories

Social Media

(DARK WEB)

A part of the Deep Web accessible only through certain browsers such as Tor designed to ensure anonymity. Deep Web Technologies has zero involvement with the Dark Web.

Illegal Information

TOR-Encrypted sites

Drug Trafficking sites

Political Protests

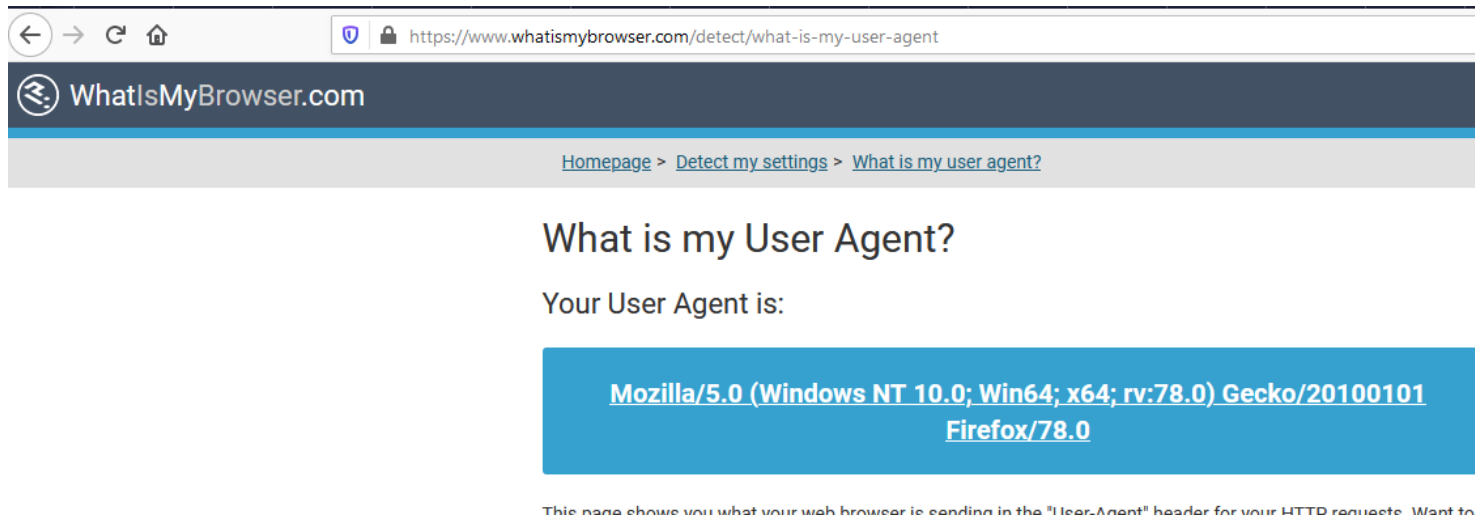
Private Communications

User agent

- When a browser or robot visits a page, it identifies itself with a `User-agent` string
 - For example, check yours out
 - [At google: "What is my user agent string"](#)
- Example from Google Chrome:
 - `Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_3) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/48.0.2564.103 Safari/537.36`
 - Previously used to indicate compatibility with the Mozilla rendering engine
 - During the "browser wars", some web sites would only send advanced features to some user agents

User agent

- You can spoof your user agent 😊
- There's nothing forcing your client software to be truthful
- User agent switcher plug in for Chrome



The screenshot shows a web browser window with the address bar containing the URL `https://www.whatismybrowser.com/detect/what-is-my-user-agent`. The page title is "WhatIsMyBrowser.com". The breadcrumb trail is "Homepage > Detect my settings > What is my user agent?". The main heading is "What is my User Agent?". Below it, the text "Your User Agent is:" is followed by a blue box containing the user agent string: `Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:78.0) Gecko/20100101 Firefox/78.0`. At the bottom, there is a small line of text: "This page shows you what your web browser is sending in the 'User-Agent' header for your HTTP requests. Want to



User agent and crawling

- Similar to a browser, when robot visits a page, it identifies itself with a User-agent
 - Just like Firefox, Chrome, Safari, etc.
 - Your site can check and behave differently if a crawler identifies itself this way
- You can request that robots not visit your site with `/robots.txt`
 - This is just a courtesy... bad crawlers can ignore it



/robots.txt

User-agent: *

Disallow: /

- User-agent: *
- means this section applies to all robots.

• Disallow: /

- tells the robot that it should not visit any pages on the site.

```
← → ↻ 🏠 Secure | https://www.defense.gov/robots.txt  
Sitemap: /SiteMap.aspx  
User-agent: *  
Disallow: *captcha*  
Disallow: /secret-human-apocalypse-plans.doc  
Disallow: /human-weaknesses-list.txt  
Disallow: /how-to-disable-any-robot.txt
```



I may have found a way to save us during the robot uprising

/robots.txt

User-agent: Googlebot-Image

Disallow: /

- Tell Google Image search not to include images from your website

/robots.txt

- Robots can ignore your /robots.txt
 - Malware robots that scan the web for security vulnerabilities
 - Email address harvesters used by spammers
- /robots.txt file is a publicly available file
 - Anyone can see what sections of your server you don't want robots to use
- /robots.txt directives can't prevent references to your URLs from other sites
 - A robot could navigate directly to a page from another website
 - e.g., if Wikipedia.org links directly to dijkstra.eecs.umich.edu/kleach/eecs485/

Agenda

- Crawler design
- **Inverted index construction**
- Deduplication
- Distributed search architecture

Serving results - speed

- After crawling is finished, we have a (big) database of documents
- To serve a search request, we need the terms in each doc

- You could just run `grep`
 - $O(N)$, where N is the total size of all web docs!
 - Too complex for fast search

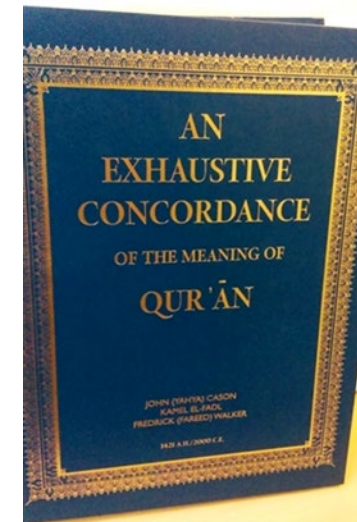
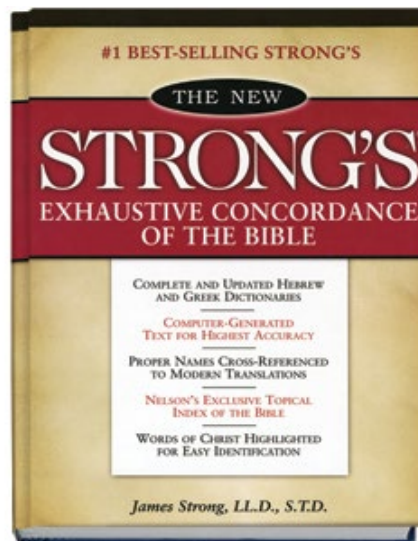
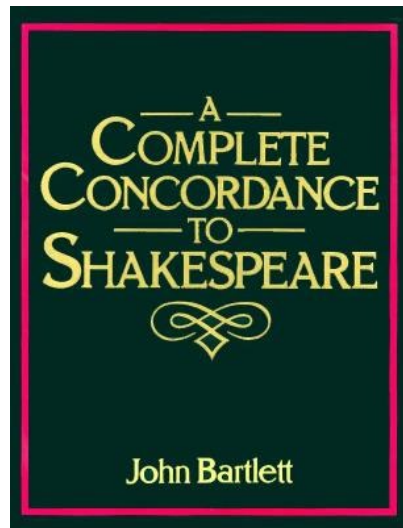
- How could we make this faster?
 - Hint: we saw it already.

Inverted index

- A *forward index* is a *list of words* in each *document*
 - Doc -> words
 - This isn't the same as a tf-idf vector
- An *inverted index* maps *words* to a *list of documents* that contain those words
 - Word -> docs
- For each word, list all the documents where that word can be found
 - Then you can do ranking on this subset
- Key to fast query processing

Inverted index

- Inverted indexes were around before computers
- Example: concordance
- List of every word, in alphabetical order
- Constructed manually before computers!!!

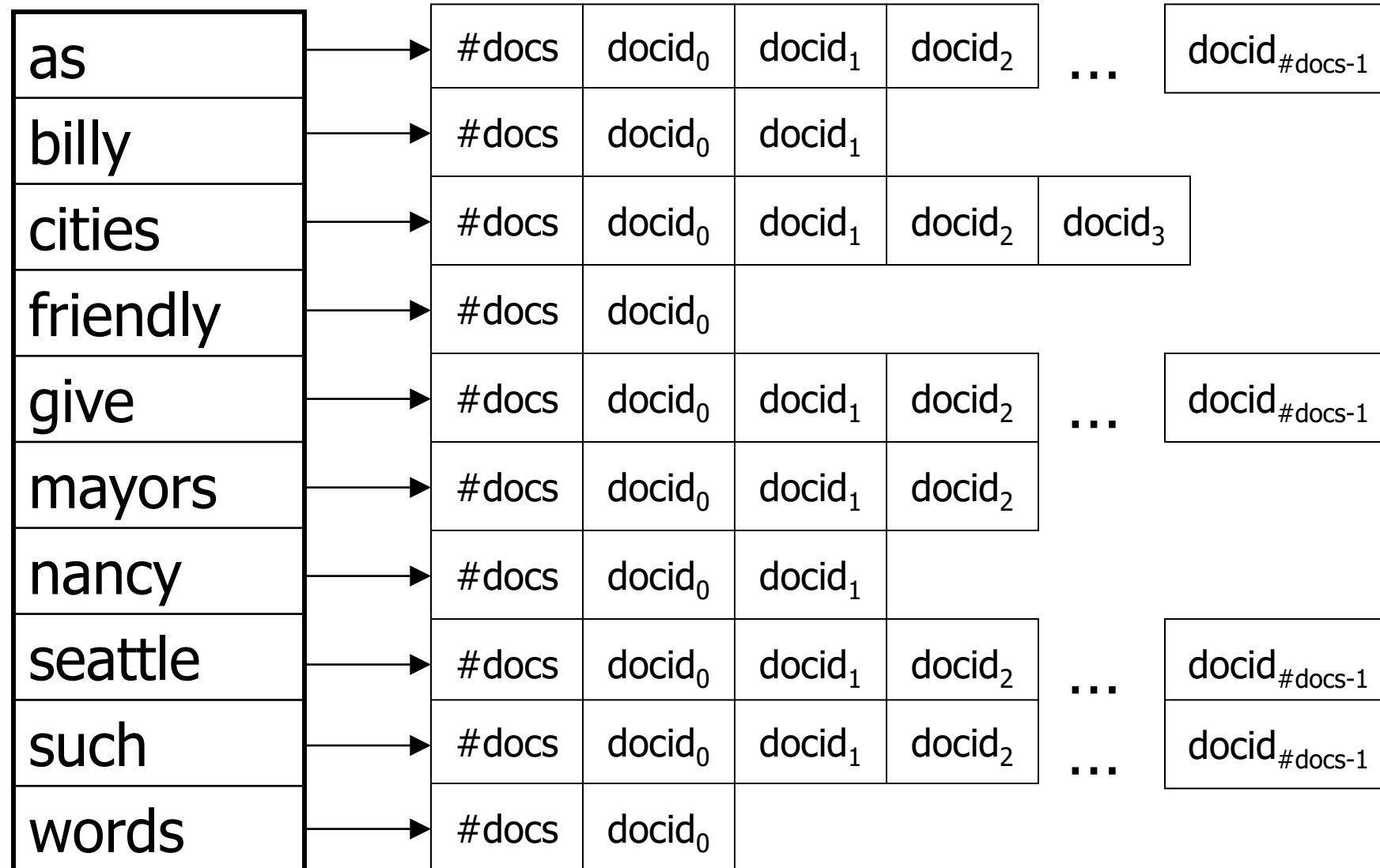


Inverted index

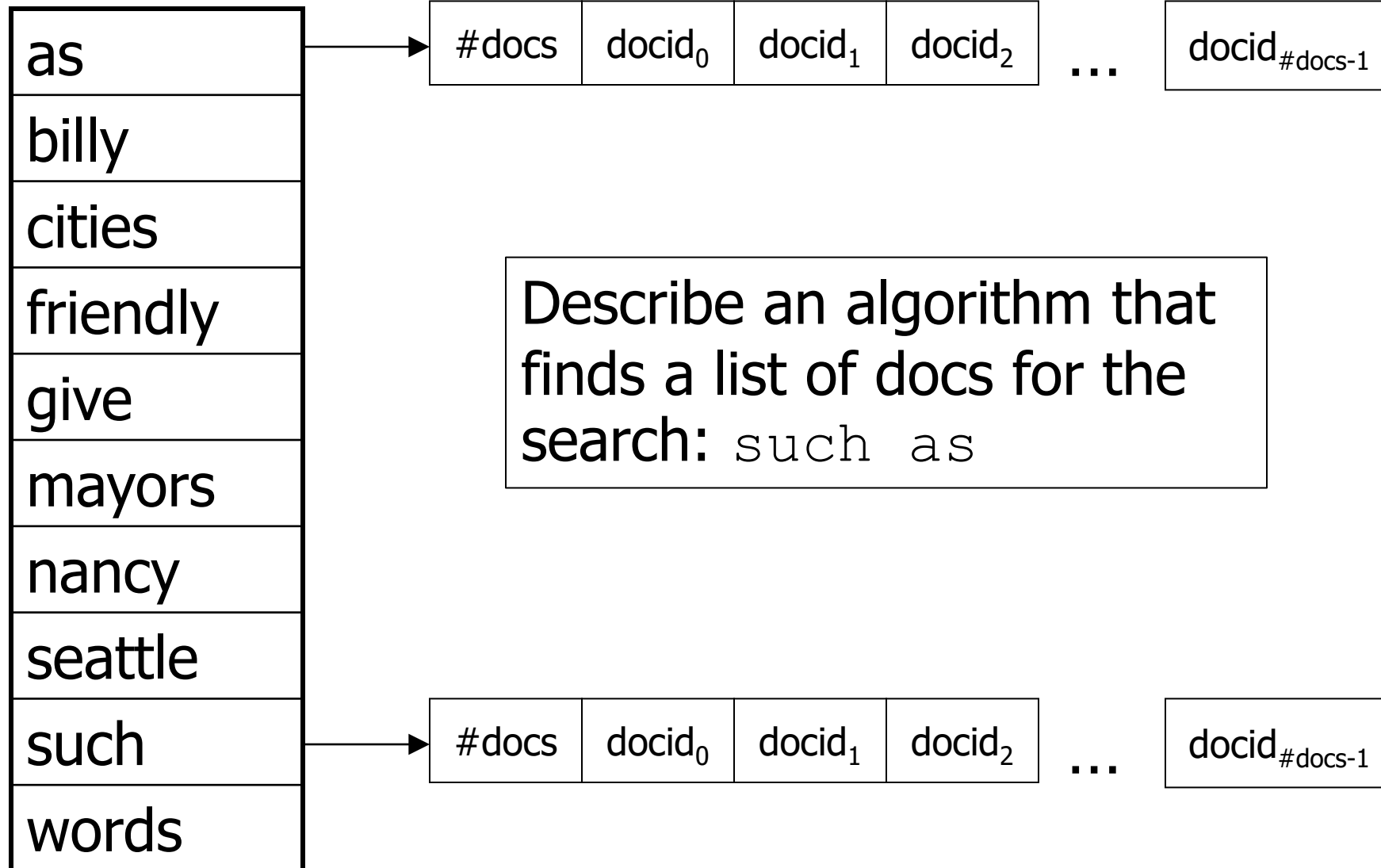
- openshakespeare.org concordance: horatio

#	Work	Character	Line	Text
1	Hamlet [I, 1]	Bernardo	13	Well, good night. If you do meet Horatio and Marcellus, The rivals of my watch, bid them make haste.
2	Hamlet [I, 1]	(stage directions)	16	Enter Horatio and Marcellus.
3	Hamlet [I, 1]	Bernardo	26	Say- What, is Horatio there ?
4	Hamlet [I, 1]	Bernardo	29	Welcome, Horatio . Welcome, good Marcellus.
5	Hamlet [I, 1]	Marcellus	32	Horatio says 'tis but our fantasy, And will not let belief take hold of him Touching this dreaded sight, twice seen of us. Therefore I have entreated him along, With us to watch the minutes of this night, That, if again this apparition come, He may approve our eyes and speak to it.
6	Hamlet [I, 1]	Marcellus	54	Thou art a scholar; speak to it, Horatio .

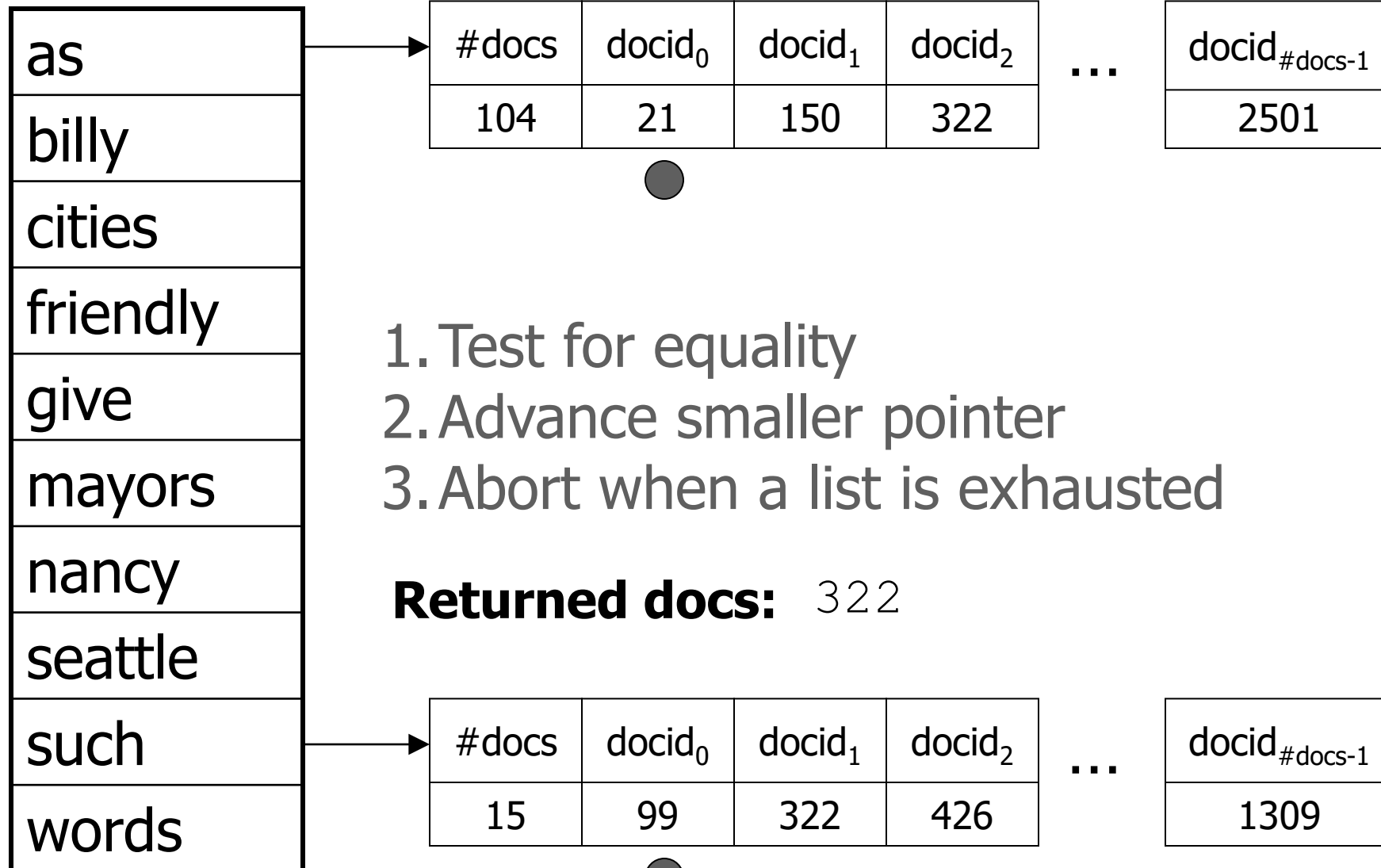
Inverted index



Inverted index exercise



Solution: merge intersection



Basic tasks

1. Compile **term-termid**, **doc-docid** maps
2. Assemble all **termid-docid** pairs
3. Sort pairs first by **termid**, then **docid**
4. Write out in *inverted-index* form

- EASY!
 - Well, not if docs won't fit into memory

Block sort-based indexing

- *External sort algorithms* work on sets larger than memory

- Block-Sort-Based Index Algorithm:

```
n = 0
```

```
while docsRemain
```

```
    n++
```

```
    block = ParseNextBlock()
```

```
    BSBI-Invert(block)
```

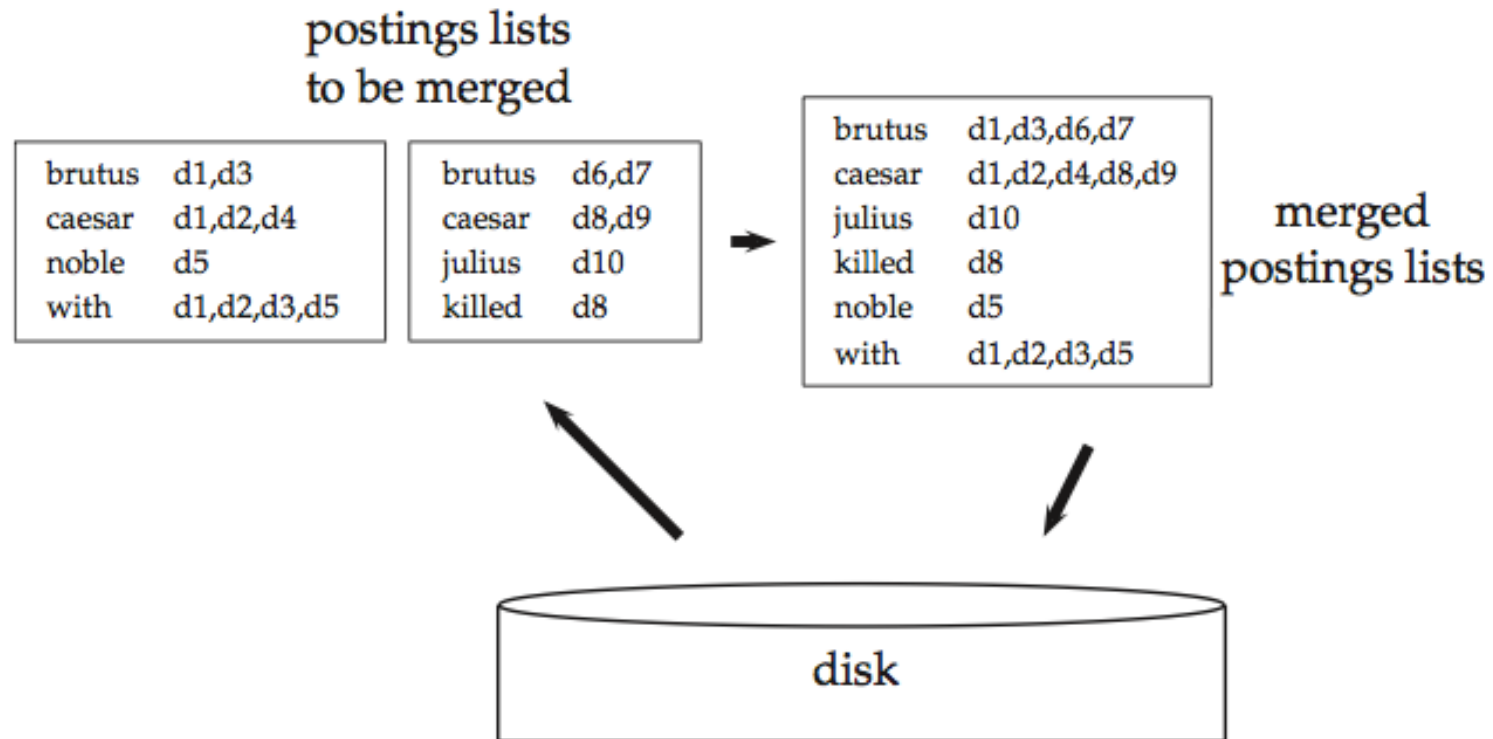
```
    WriteToDisk(block,  $f_n$ )
```

```
MergeBlocks( $f_1, \dots, f_n$ ) =>  $f_{\text{merged}}$ 
```

Block sort-based indexing

- `ParseNextBlock` accumulates termid-docid pairs in memory until block is full
- `BSBI-Invert` generates small in-memory inverted index
- So: we build a series of small in-memory inverted indexes, writing each one to disk
- Finally: we merge them

Block merging

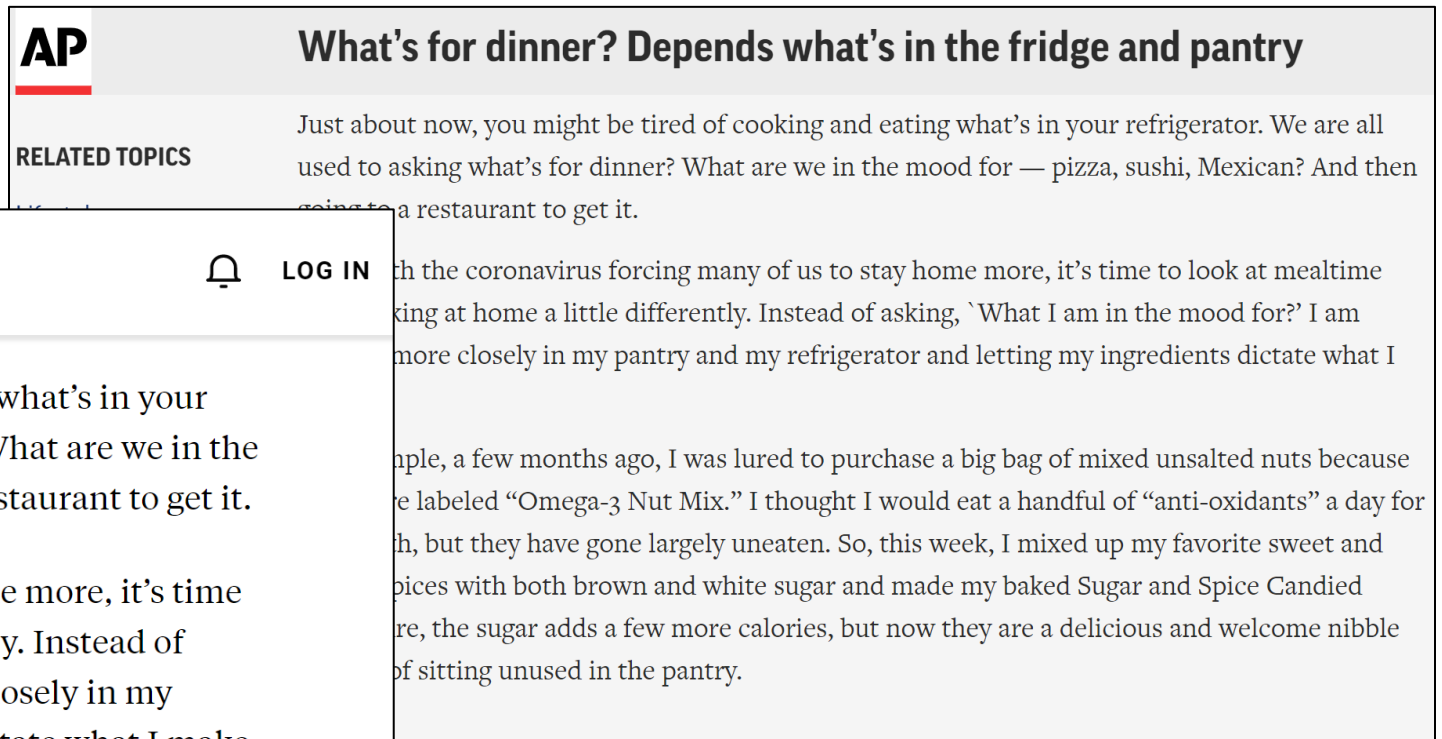


Agenda

- Crawler design
- Inverted index construction
- **Deduplication**
- Distributed search architecture

Deduplication

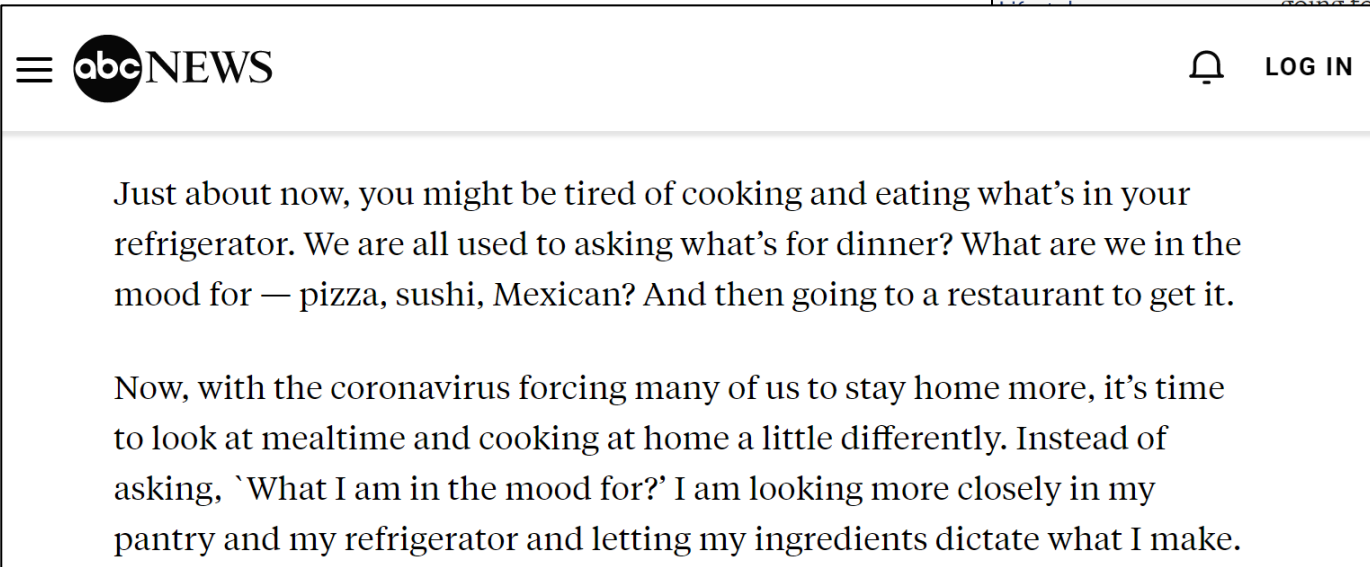
- How can you be sure a web page is worth indexing?
 - Has it changed meaningfully compared to previous version?
 - Is it just a clone of another site? (weirdly common)



AP **What's for dinner? Depends what's in the fridge and pantry**

Just about now, you might be tired of cooking and eating what's in your refrigerator. We are all used to asking what's for dinner? What are we in the mood for — pizza, sushi, Mexican? And then going to a restaurant to get it.

RELATED TOPICS



abc NEWS **LOG IN**

Just about now, you might be tired of cooking and eating what's in your refrigerator. We are all used to asking what's for dinner? What are we in the mood for — pizza, sushi, Mexican? And then going to a restaurant to get it.

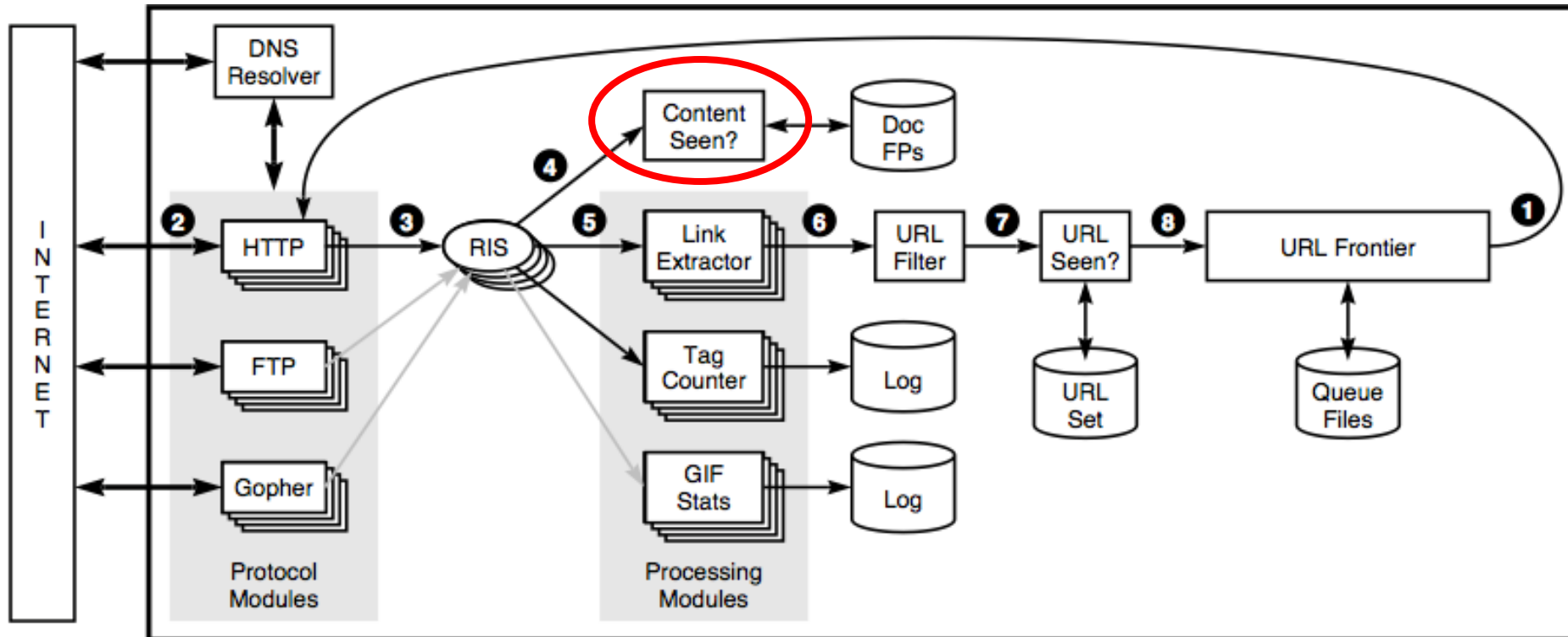
Now, with the coronavirus forcing many of us to stay home more, it's time to look at mealtime and cooking at home a little differently. Instead of asking, 'What I am in the mood for?' I am looking more closely in my pantry and my refrigerator and letting my ingredients dictate what I make.

With the coronavirus forcing many of us to stay home more, it's time to look at mealtime cooking at home a little differently. Instead of asking, 'What I am in the mood for?' I am looking more closely in my pantry and my refrigerator and letting my ingredients dictate what I make.

For example, a few months ago, I was lured to purchase a big bag of mixed unsalted nuts because they were labeled "Omega-3 Nut Mix." I thought I would eat a handful of "anti-oxidants" a day for my health, but they have gone largely uneaten. So, this week, I mixed up my favorite sweet and spicy cookies with both brown and white sugar and made my baked Sugar and Spice Candied Pecans. Sure, the sugar adds a few more calories, but now they are a delicious and welcome nibble instead of sitting unused in the pantry.

Deduplication

- Deduplication in Mercator



Deduplication

- Two problems to solve
 1. Are these two documents duplicates?
 - Shingling
 - Jaccard similarity coefficient
 2. Find all duplicates

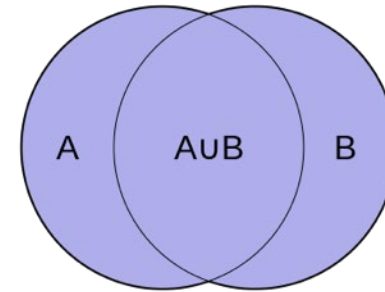
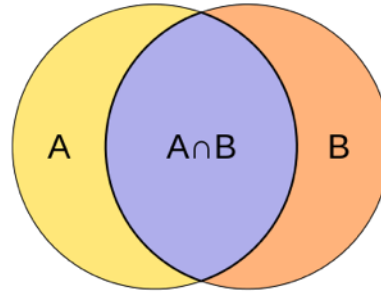
Shingling

- Compute the *k-shingles* for a page
- “I think EECS 485 is a great class”
 - For $k=3$: “I think EECS”, “think EECS 485”, “EECS 485 is”, etc.
- If two docs share lots shingles, they’re duplicates
- Each document is now a *set* of shingles
- Why might you pick larger or smaller values of k ?
- We now have a set comparison problem
 - How similar are the two sets of *k-shingles*?

Jaccard similarity coefficient

- Jaccard similarity coefficient compares the similarity of the two sets of shingles (A and B)
- Size of the intersection / size of the union

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$



- 0 for disjoint sets, 1 for equal sets
- What is the complexity of computing Jaccard?
- Assume A and B are size $O(N)$

Jaccard similarity coefficient

- What is the complexity?
- Assume A and B are size O(N)

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

- O(N) time with O(N) space using hash tables
- More precisely, it's O(|A| + |B|)

Deduplication

- We now have a way to answer "are these two documents duplicates?"
 - Shingling
 - Jaccard similarity coefficient
- Speed up Jaccard similarity coefficient computation by approximating it

Jaccard similarity coefficient

- Computing the Jaccard similarity coefficient is slow for large documents
- Can we compute this answer without explicitly computing the full union and intersection?
- **Let's estimate it!**
- First, a question:
- Pick a random shingle from $A \cup B$
- What is the probability that this shingle is in the intersection?

Jaccard similarity coefficient

- Pick a random shingle from $A \cup B$
- What is the probability that this shingle is in the intersection?

$$\frac{|A \cap B|}{|A \cup B|}$$

- That's the same as the Jaccard similarity coefficient

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

- If we can estimate the probability of a random shingle being in the intersection, we can efficiently approximate Jaccard similarity coefficient

Selecting shingles

- How can we efficiently select a random shingle that is present in at least one of A or B?
 - In the set $A \cup B$
- Use the **MinHash** algorithm



MinHash intuition

$a = \{ \text{"hello"}, \text{"the"}, \text{"quick"} \}$

$b = \{ \text{"quick"}, \text{"brown"}, \text{"fox"} \}$

$a \cup b = \{ \text{"hello"}, \text{"the"}, \text{"quick"}, \text{"brown"}, \text{"fox"} \}$

Similarity: $\frac{|\text{hello}|}{|\text{hello,the,quick,brown,fox}|} = \frac{1}{5}$

If we shuffle $a \cup b$, what is the probability that “quick” comes first?

(it's also 1/5)

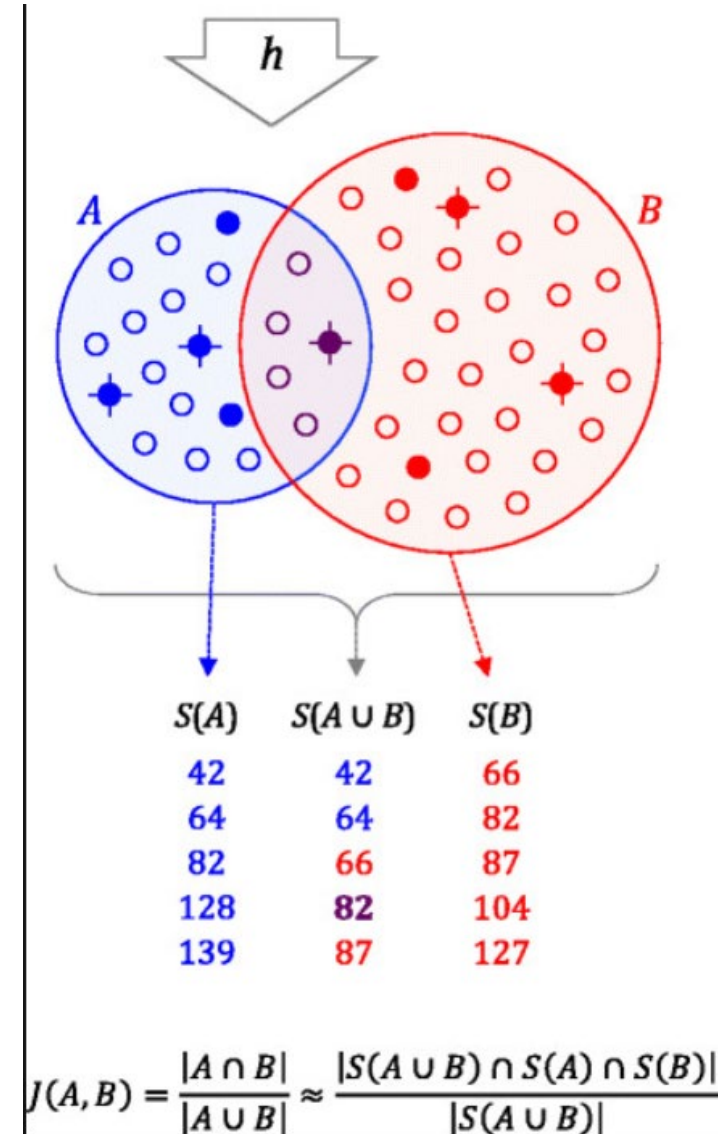
Intuition: hashing is like a random shuffle... let's hash the elements and find if the smallest hash matches

MinHash: Fixed-time Jaccard estimation

- Hash each shingle in set A , pick element with **smallest hash value**
 - The hash function maps inputs uniformly over the output
 - Selecting the $\min(h(x))$ is the same as selecting a random item x !
 - (with a hash function like SHA256)
- Find minimum hashes $h_{\min}(A)$ and $h_{\min}(B)$
- Why? “Probability magic”
 - Among all $A \cup B$, if an element is in $A \cap B$, then two elements would share the same hash
 - AND: The probability of the element in $A \cap B$ having the *minimum* hash is **exactly equal to** the Jaccard Similarity
- Apply this with k different hashing algorithms

MinHash idea

- Why MinHash?
- Computing hash values for a set is $O(N)$
 - For fixed k
 - Sometimes called computing the “signature” of the set
 - Compute this once
- Comparing two sets is now constant time (fixed k)
 - Do this many times when crawling or indexing



Comparing using signatures

document signatures

[0x0b1a, 0x3dda, 0x43cf] => score: 1/3

[0x6c4f, 0x3222, 0x232a] => score: 2/3

comparison document signature

[0x0b1a, 0x3222, 0x232a]

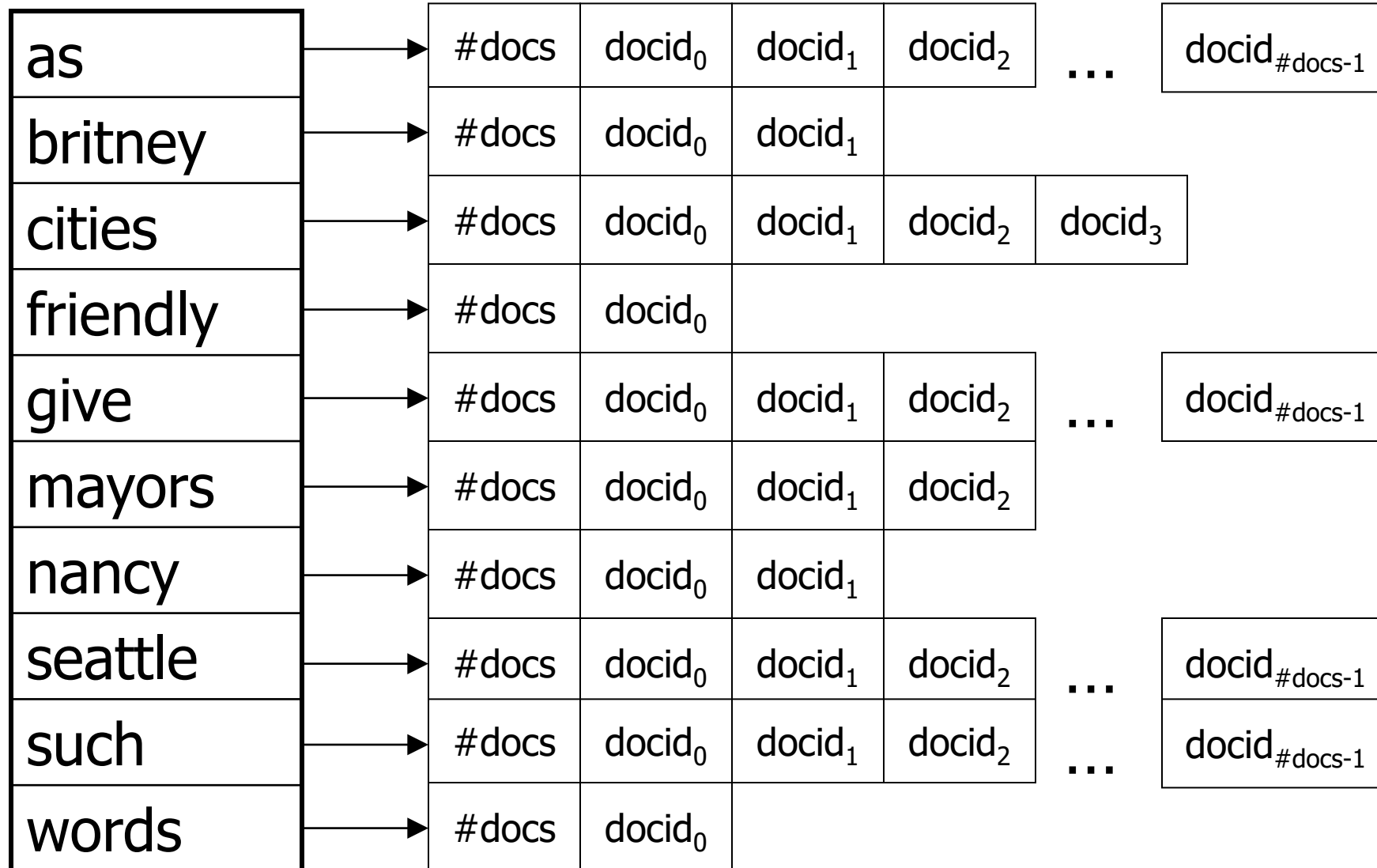
Agenda

- Crawler design
- Inverted index construction
- Deduplication
- **Distributed search architecture**

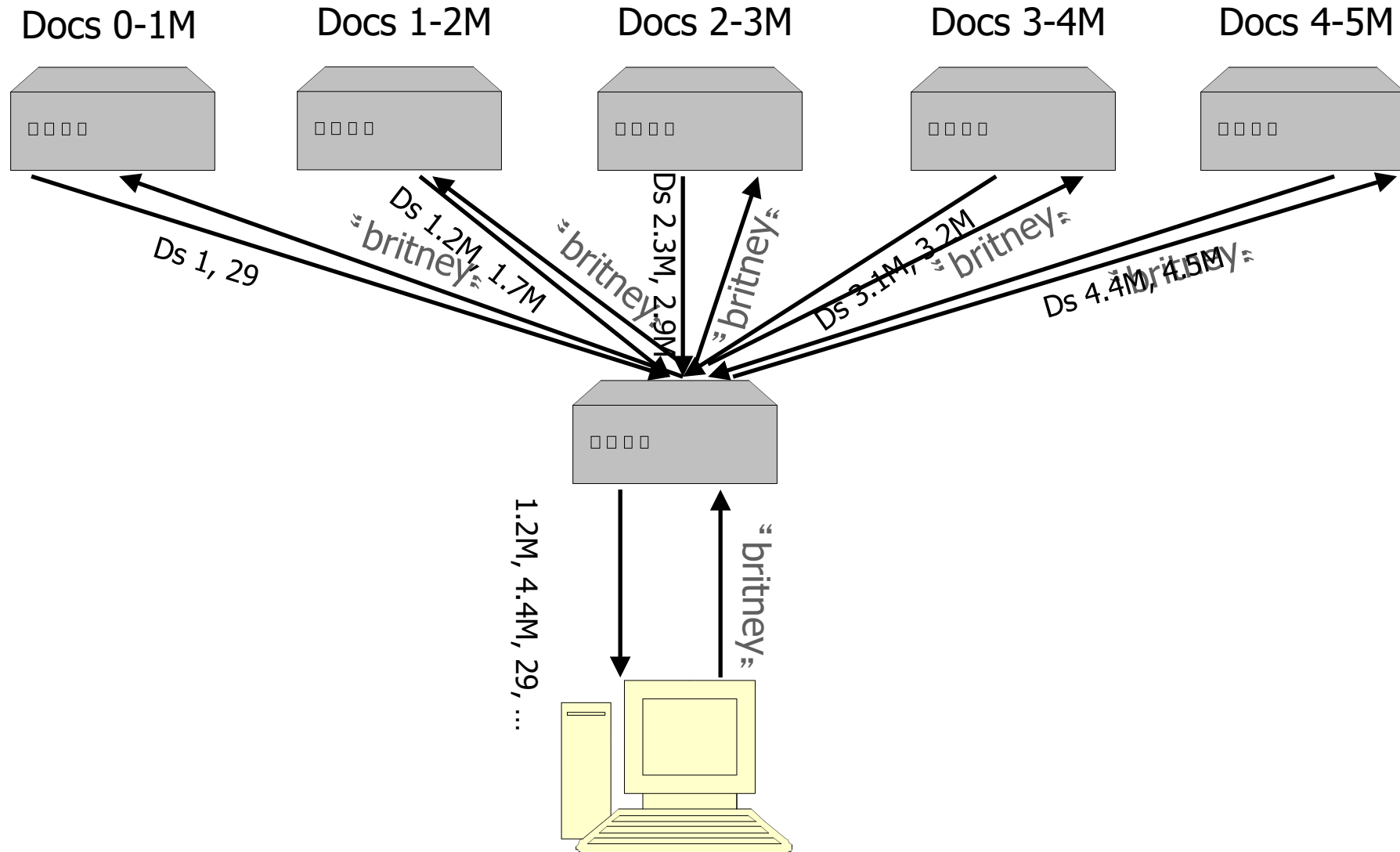
Distributed searching

- Not even the inverted index is small enough for one machine to handle it
 - Billions of docs
 - Hundreds of millions of queries
- Also, what if the machine fails?
- Need to parallelize query processing
 - Segment by document
 - Segment by search term

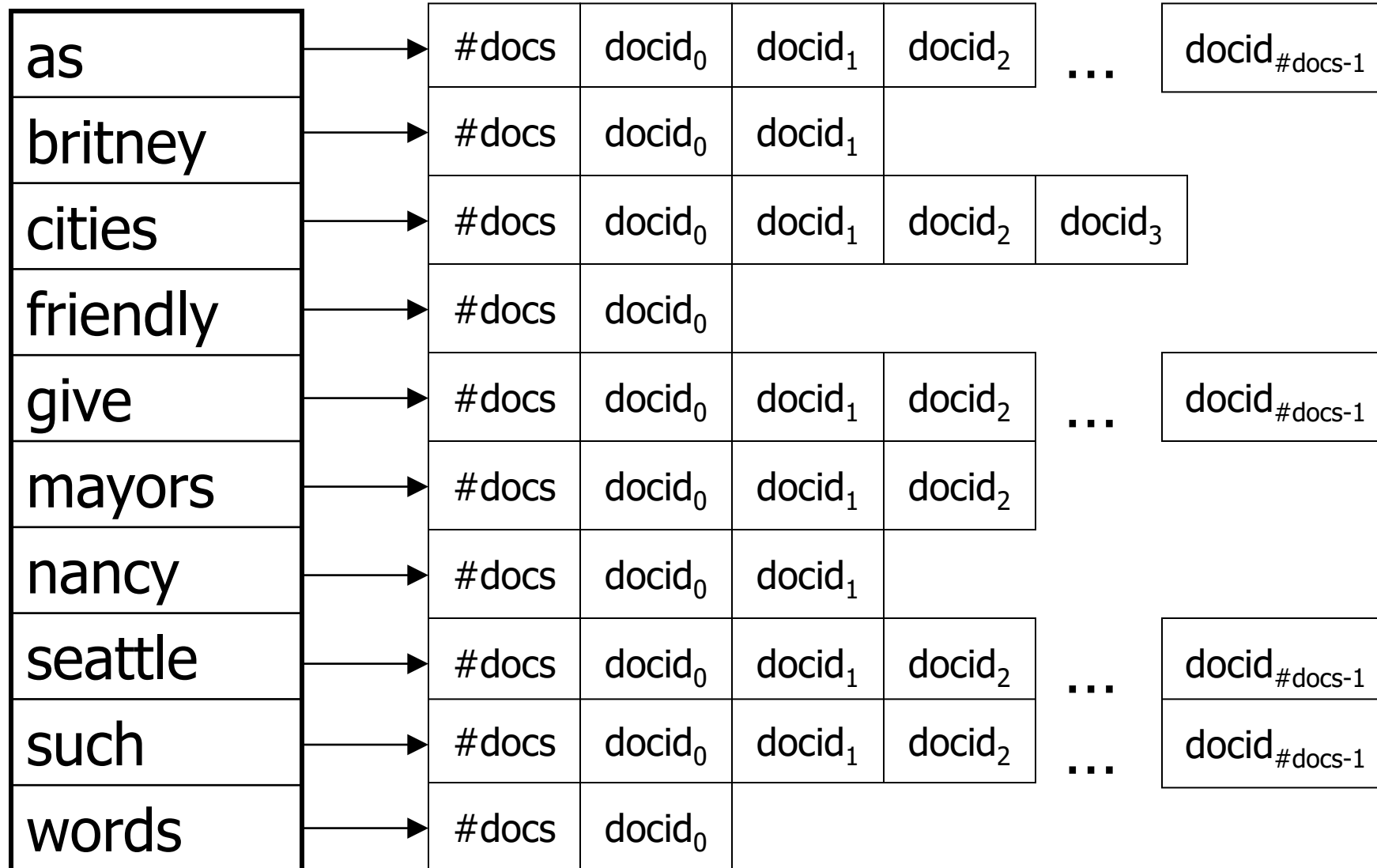
Segment by document (divide cols)



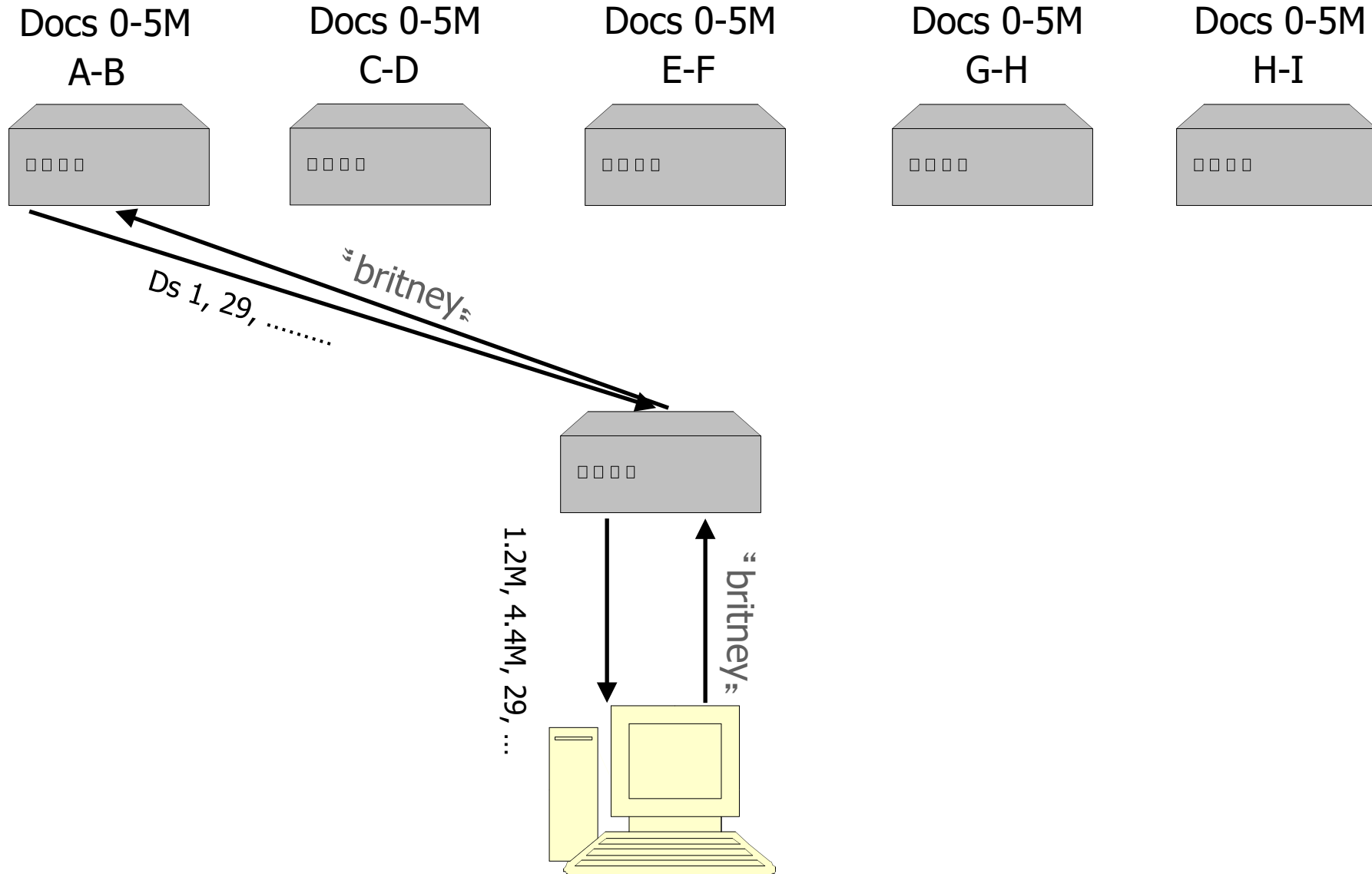
Segment by document



Segment by term (divide rows)



Segment by term



Segmentation

- Segment by document
 - Easy to partition (just MOD the docid)
 - Easy to add new documents
 - If machine fails, quality goes down but queries don't die
- Segment by term
 - Harder to partition (terms uneven)
 - Trickier to add a new document (need to touch many machines)
 - If machine fails, search term might disappear, but not critical pages (e.g., cnn.com/index.html)