NYU TANDON SCHOOL OF ENGINEERING

CS6913: Web Search Engines

Torsten Suel

CSE Department

NYU Tandon School of Engineering

torsten.suel@nyu.edu

What is this Course About ?

- learn how search engines work   (google, bing, baidu)
- learn about recent developments in search (incl. NN-based search)
- learn about the field of Information Retrieval (IR)
- learn about data compression & computing with large data (a little)
- learn about current research challenges in the field
- learn how to build search tools!
  - basic information retrieval techniques
  - what software tools to use
  - system architectures and performance
  - how to work with GBs or TBs of data

NYU | TANDON SCHOOL OF ENGINEERING

Not the Focus of this Course:

- web site design and HTML, javascript, etc
- building web applications, PHP scripting etc.
- how to use search engines

Not the main topic, but will come up during the course:

- machine learning and LLMs
- data analysis & data serving systems (hadoop, spark, GFS, ...)
- image and multimedia search
- social network analysis
- natural language processing (NLP)
- recommender systems

NYU | TANDON SCHOOL OF ENGINEERING

Lecture 1: Introduction

I. The Web and Web Search:

- how the web works
- search tools

II. Introduction to Information Retrieval

- origins of IR
- basic setups and techniques

III. Main Components of a Search Engine

- basic search engine architecture
- web crawling
- indexing
- querying and ranking

NYU | TANDON SCHOOL OF ENGINEERING

Lecture 1: Introduction

I. The Web and Web Search:

- how the web works
- search tools

II. Introduction to Information Retrieval

- origins of IR
- basic setups and techniques

III. Main Components of a Search Engine

- basic search engine architecture
- web crawling
- indexing
- querying and ranking

I. The Web and Web Search:

How the web works:

- **Three different views of the web:**
  - text view
  - graph view
  - site and domain structure

- **Client-server paradigm of the web**
  - urls, htmls, and http
  - dns: domain name service

NYU TANDON SCHOOL OF ENGINEERING

Text View of the Web: “Collection of Documents”

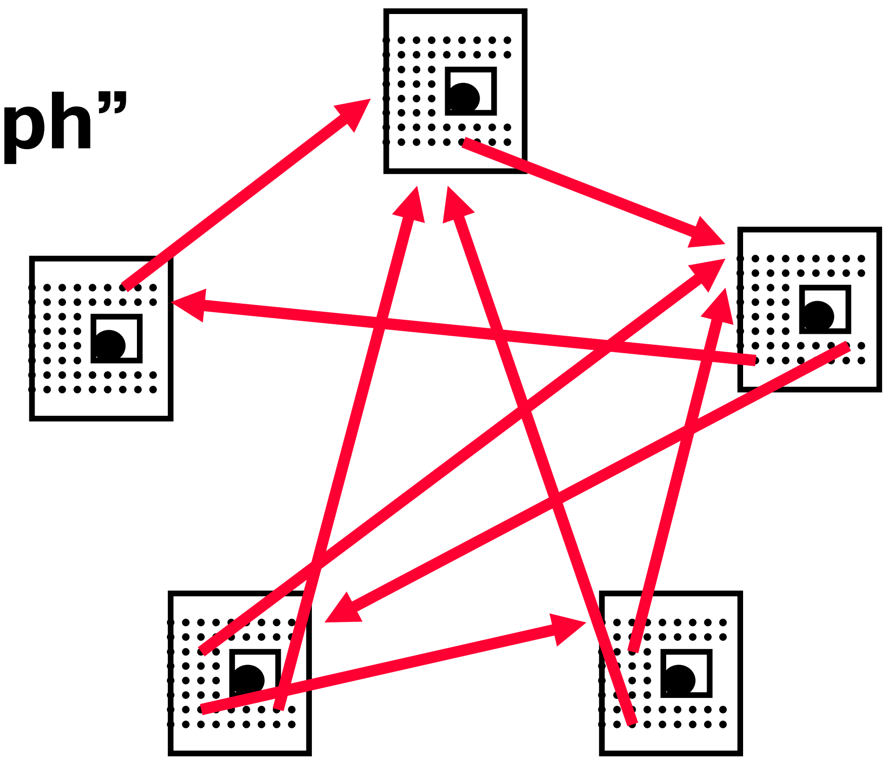
- billions of pages of text

- with html markup
- and flash, javascript, etc
- and images, audio, etc
- and cgi
- and database backends

- but basically, text
- “pages” or “documents”

NYU | TANDON SCHOOL OF ENGINEERING

Graph View: "The Web Graph"



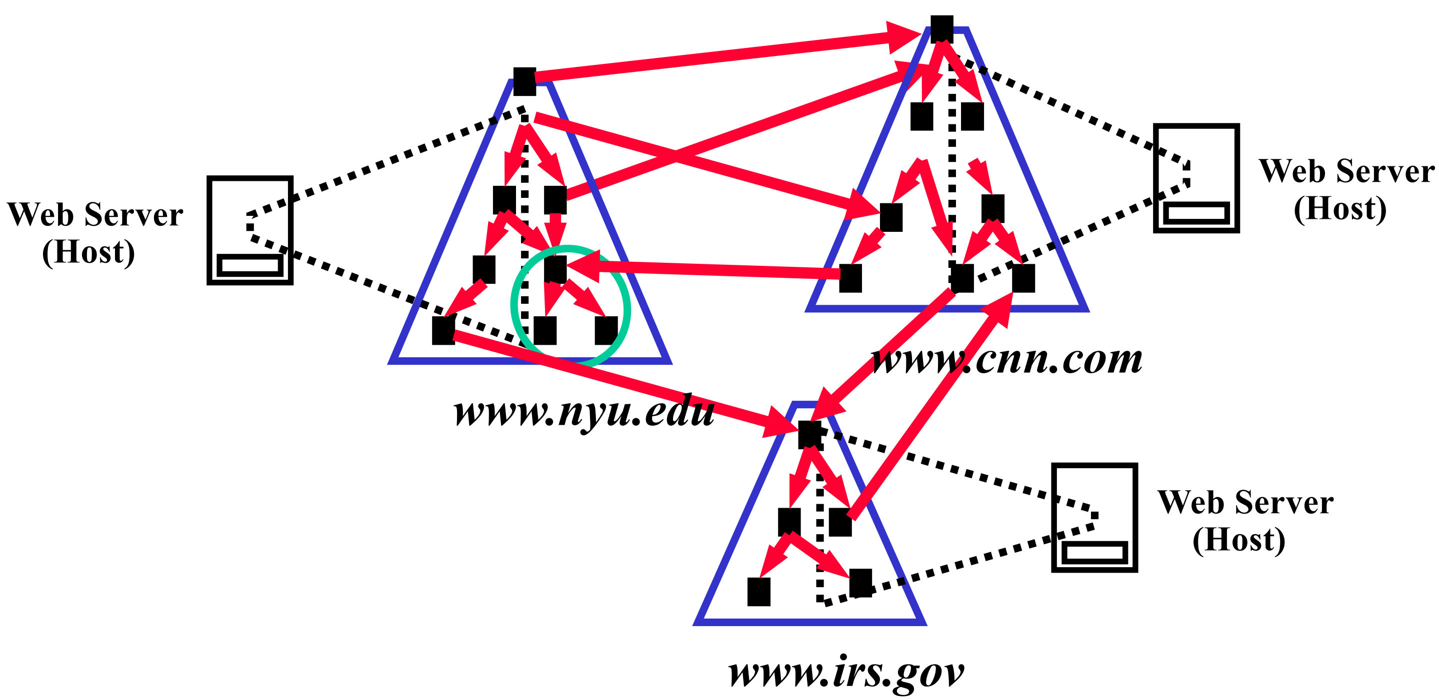
- hypertext, not just text
- links between pages
- hundreds of billions of links
- the web is a giant graph!

- hyperlinks have meaning
- many links to a page → good page?
- old idea in citation and social network analysis

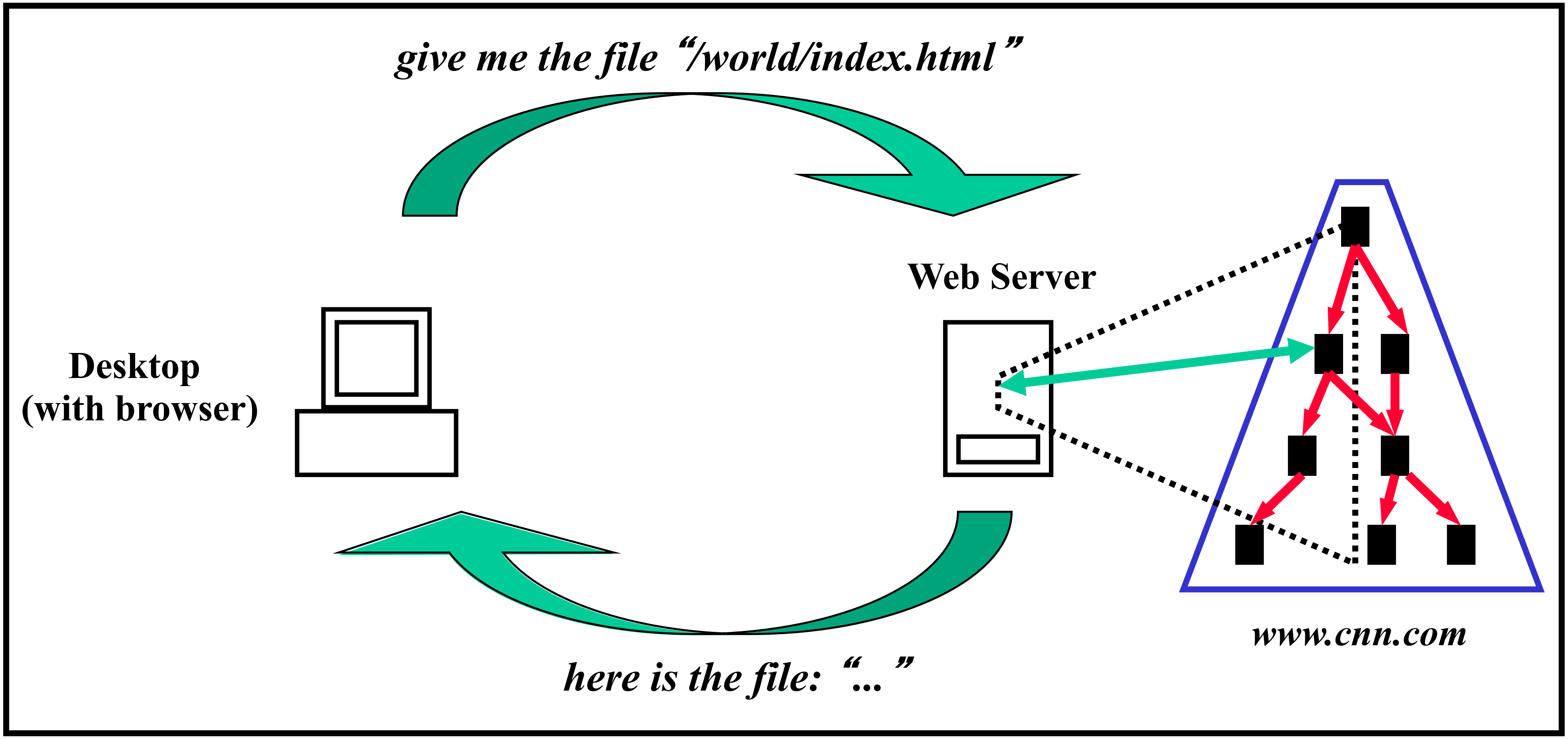
- local and global links
- page graph vs. site graph

Site and Domain Structure: Hierarchical View

- domain structure: .edu .nyu.edu .cse.nyu.edu
- internal structure: folders/directories
- folders, sites, and domains often contain related pages
- induces a natural topical clustering of the collection
- also, menu structure and boilerplate detection important



How Web Access Works:



Three Main Ingredients:

- **Naming: URL (uniform resource locators)**
  (used to identify and locate objects)
- **Communication: HTTP (hypertext transfer protocol)**
  (used to request and transfer objects)
- **Rendering: HTML (hypertext markup language)**
  (used to defined how object should be presented to user)

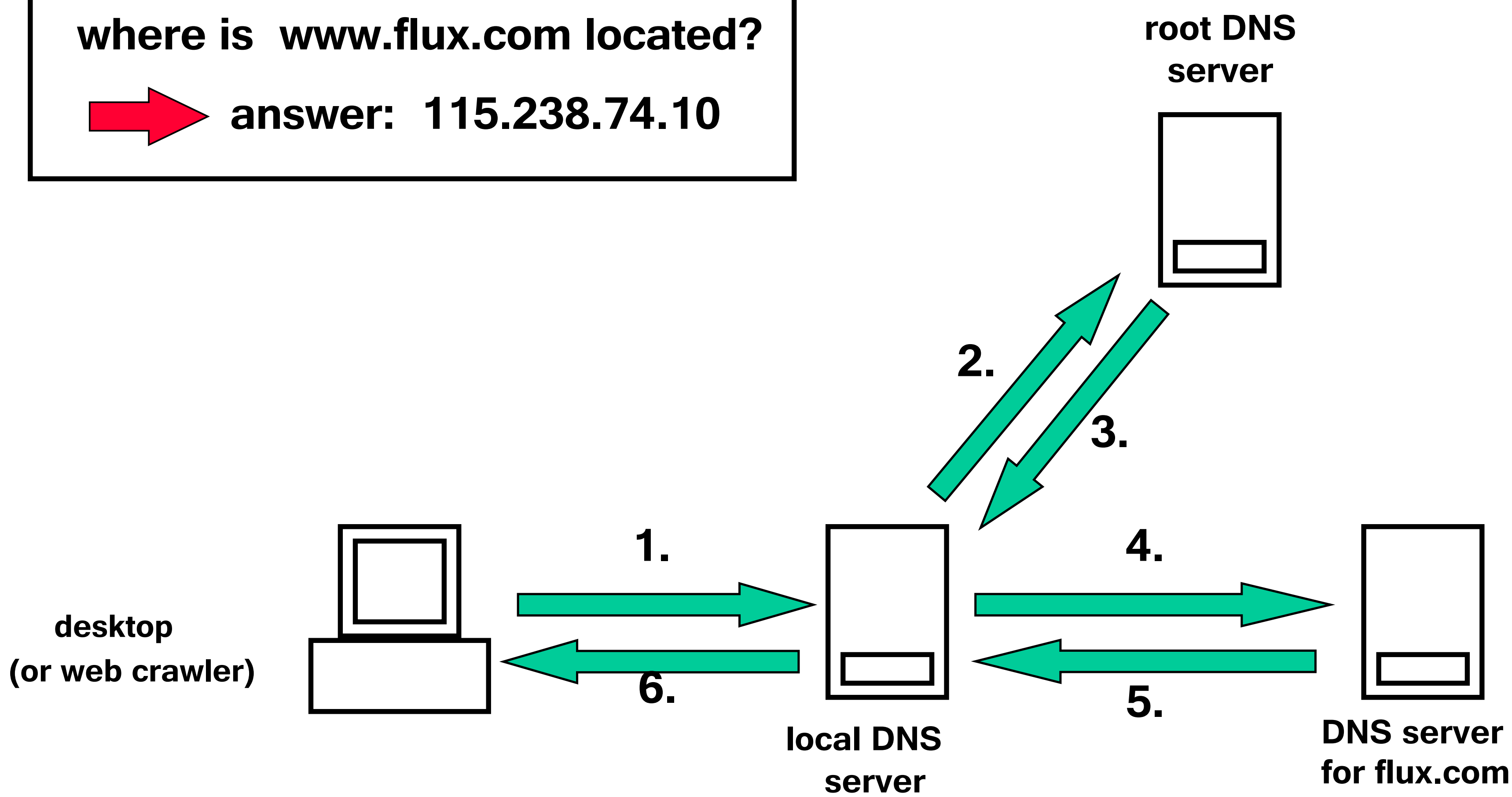
Client Server Paradigm:

- **Browser uses HTTP to ask web server for an object identified by a URL, and renders this object according to rules defined by HTML**

Domain Name Service:

where is www.flux.com located?

answer: 115.238.74.10



Names, Addresses, Hosts, and Sites

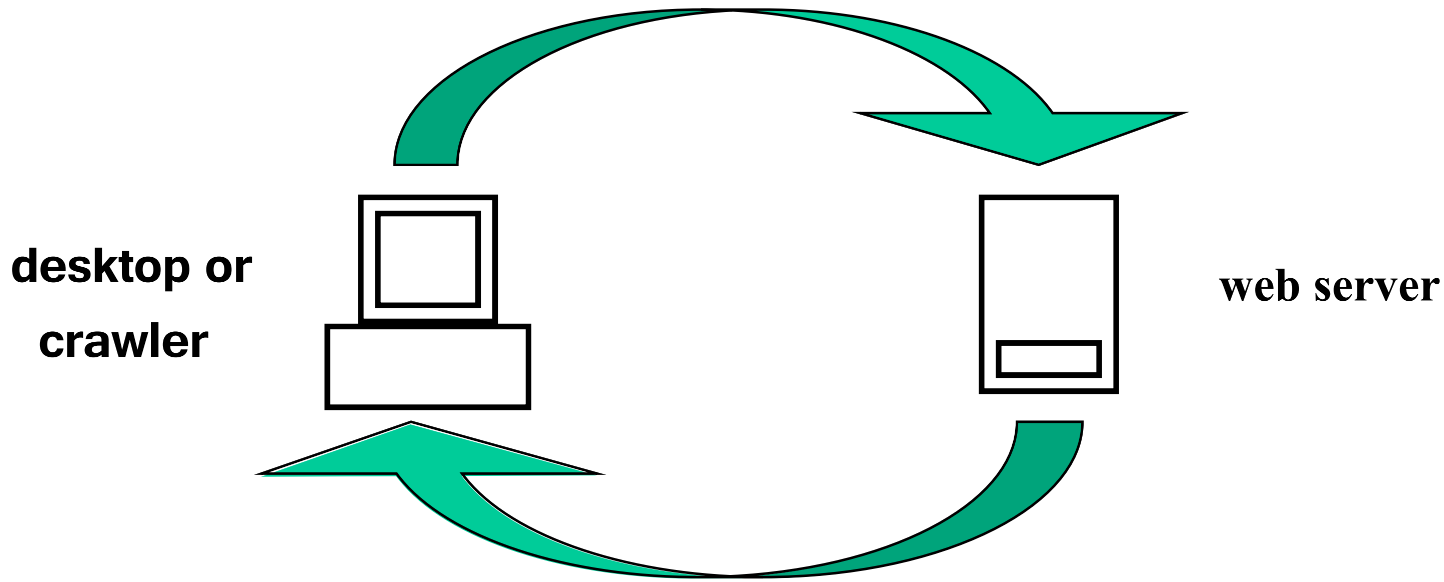
- a machine can have several host names and IP addresses
- a host can have many sites (what is a site?)
- a site can be served by many hosts, or by CDNs
  (content distribution networks)
- issues: detecting duplicates, crawling, local vs. global links

```
weasel% nslookup www.cnn.com
Server:  photon.poly.edu
Address:  128.238.32.22

Non-authoritative answer:
Name:    cnn.com
Addresses:  207.25.71.25, 207.25.71.26, 207.25.71.27, 207.25.71.28
          207.25.71.29, 207.25.71.30, 207.25.71.5, 207.25.71.6, 207.25.71.20
          207.25.71.22, 207.25.71.23, 207.25.71.24
Aliases:  www.cnn.com
```

HTTP:

GET /world/index.html HTTP/1.0
User-Agent: Mozilla/3.0 (Windows 95/NT)
Host: www.cnn.com
From: …
Referer: …
If-Modified-Since: ...

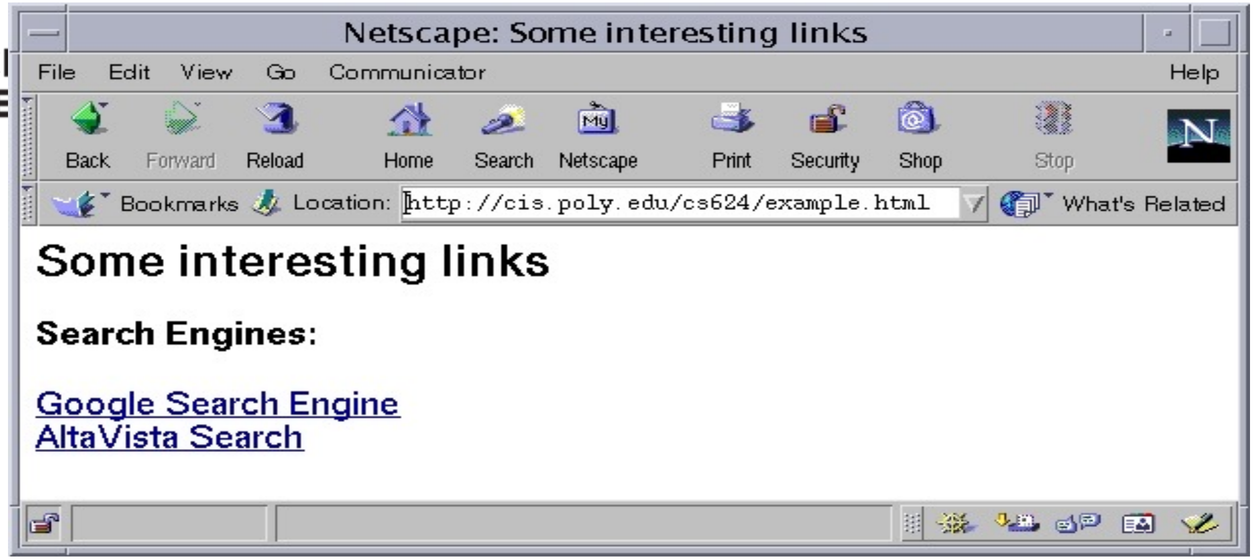


HTTP/1.0 200 OK
Server: Netscape-Communications/1.1
Date: Tuesday, 8-Feb-99 01:22:04 GMT
Last-modified: Thursday, 3-Feb-99 10:44:11 GMT
Content-length: 5462
Content-type: text/html

<the html file>

NYU TANDON SCHOOL OF ENGINEERING

HTML:




```
<HTML>
<HEAD>
   <TITLE> Some interesting links </TITLE>
</HEAD>
<body BGCOLOR="#FFFFFF" LINK="#0000FF" VLINK="000099">

<H1>
Some interesting links
</H1>

<p>
<strong>Search Engines:</strong><p>
<a href="http://www.google.com">Google Search Engine</a><br>
<a href="http://www.altavista.com">AltaVista Search</a>

</BODY>
</HTML>
```

Challenges for Search Engines due to HTTP/HTML

- complex URLs:

  http://www.google.com/search?q=brooklyn

  http://www.amazon.com/exec/obidos/ASIN/1558605703/qid%3D9...

  http:/cis.poly.edu/search/search.cgi

- result page can be computed by server in arbitrary manner!
- file types: mime types and extensions
- automatic redirects
- cookies and sessions
- javascript/flash/activeX etc technologies
- content distribution networks (CDNs), advertising networks

Things have gotten more and more complex:

- many web sites are just frontends to databases/programs
- sites such as facebook, linkedIn, twitter not fully crawlable
  - to protect user privacy
  - to preserve economic advantage for the site
  - because crawling would be expensive or impossible
- advertising networks and user profiling (instrumentation)
- personalization of content
- web spam and auto-generated content

- this is not the web of 1995!
- it is not just about "fetching files" anymore

Lecture 1: Introduction

I. The Web and Web Search:

- how the web works
- search tools

II. Introduction to Information Retrieval

- origins of IR
- basic setups and techniques

III. Main Components of a Search Engine

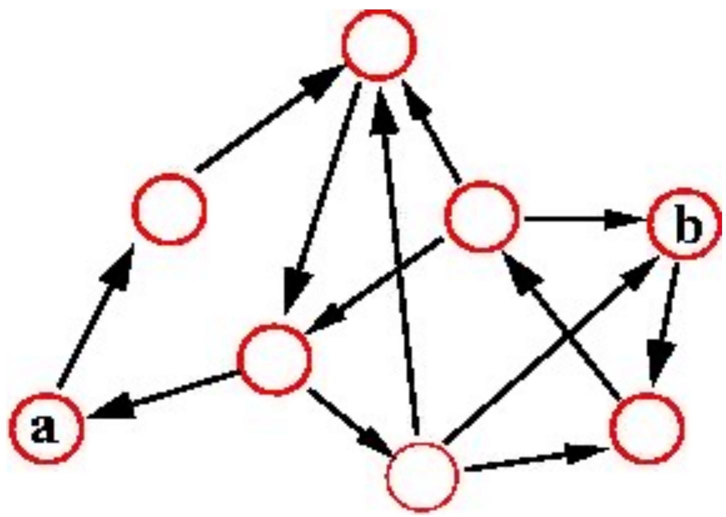
- basic search engine architecture
- web crawling
- indexing
- querying and ranking

Motivation: Finding Pages/Info on the Web?

- 130 trillion pages: 1.3 * 10^14   (Google 2016)
- hundreds of trillion of hyperlinks
- plus images, movies, .. , database content
- IR paradigms: searching versus browsing
- browsing does not work well on the web   (with some exceptions)

→ we need search tools for finding pages and information

→ search engines and related tools

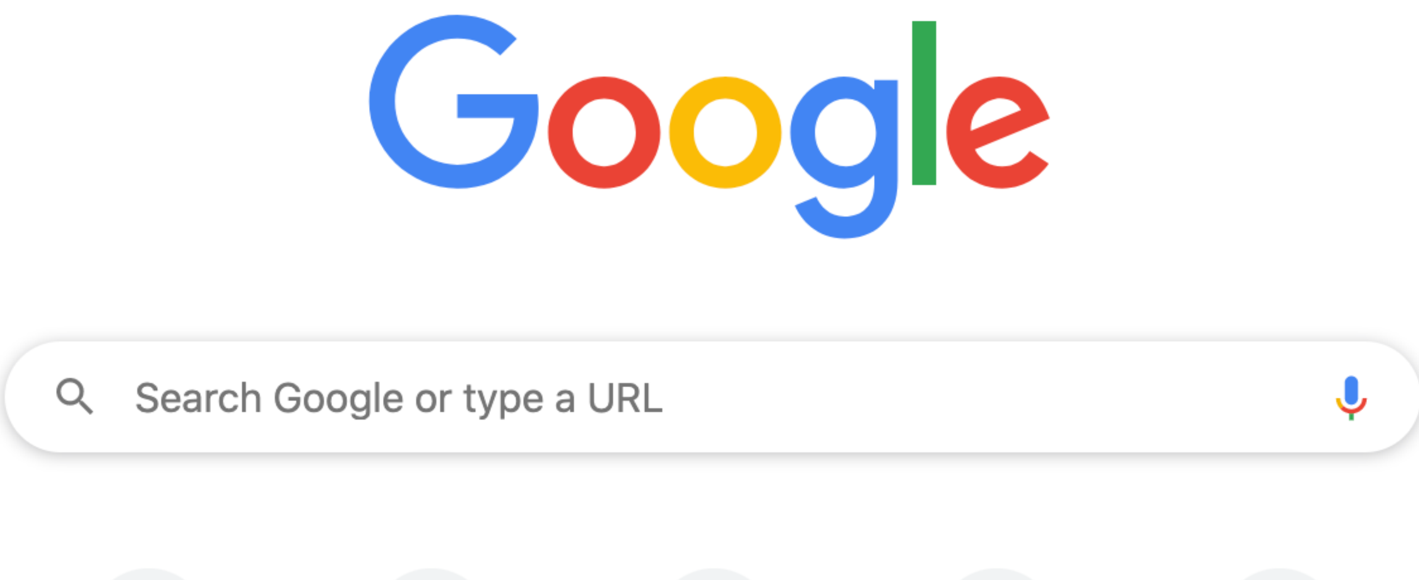


Overview of Search Tools and Scenarios

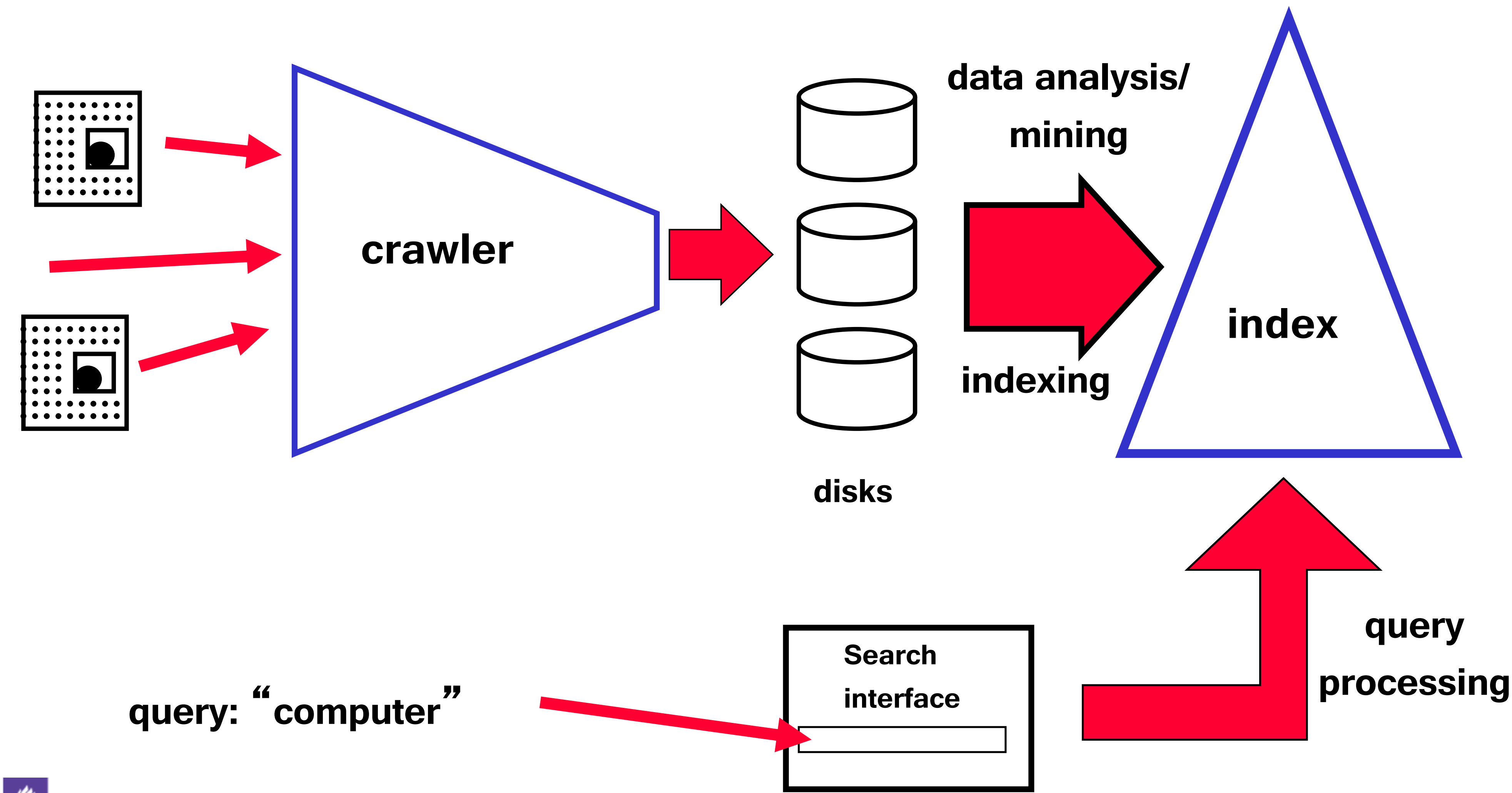
- **major search engines** *(google, bing, baidu, yandex)*

- **specialized search engines** *(news, legal, medical)*
- **social network search** *(facebook, linkedin)*
- **e-commerce search** *(amazon, google shopping, ebay)*
- **sponsored search and ad placement**
- **recommender systems** *(amazon, netflix, spotify)*
- **question answering, assistants, and chat bots**
- **multimedia search – images, video, audio**
- **privacy-preserving search**

Major Search Engines:



Basic Structure of a Search Engine:



Four Major Components:

- data acquisition / web crawling
  - to collect pages from all over the web
  - messy process with lots of hand tuning
- data analysis / web mining
  - link analysis, spam detection, query log analysis
  - using mapreduce or similar specialized platforms
- index building and maintenance
- query processing / result ranking

most of the cycles spent in data analysis and query proc.

NYU | TANDON SCHOOL OF ENGINEERING

Ranking:

Advanced Search   Preferences   Search Tips

Google

computer   [Google Search]   [I'm Feeling Lucky]

Searched the web for **computer** .   Results **1 - 10** of about **35,400,000**. Search took **0.**

COMPUTERS - Lowest Prices on all Computers!   Spo
www.BizRate.com   **Click Here to Stop Searching and Start SHOPPING!**

Category: Computers > Computer Science > Journals

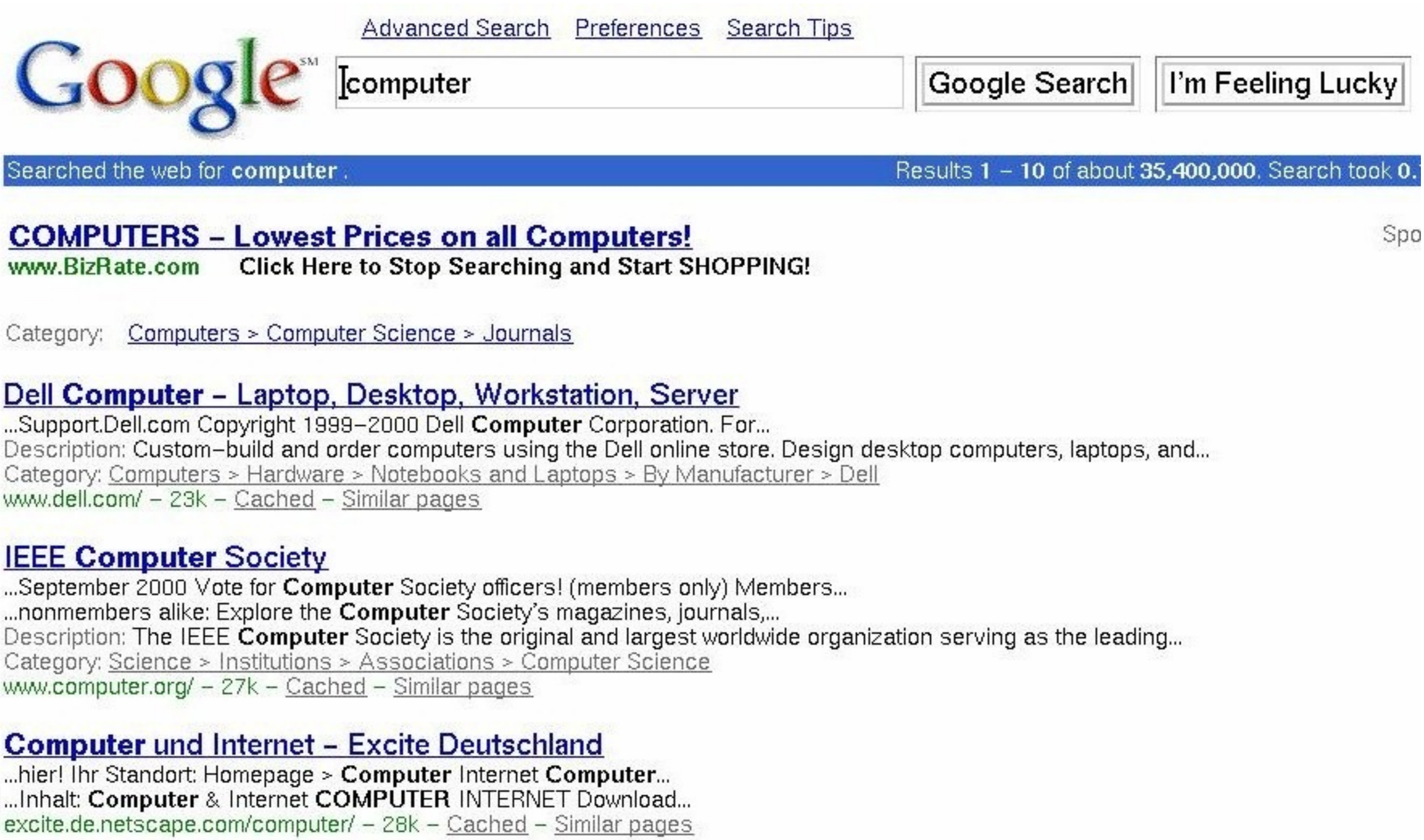
Dell Computer - Laptop, Desktop, Workstation, Server
...Support.Dell.com Copyright 1999-2000 Dell **Computer** Corporation. For...
Description: Custom-build and order computers using the Dell online store. Design desktop computers, laptops, and...
Category: Computers > Hardware > Notebooks and Laptops > By Manufacturer > Dell
www.dell.com/ - 23k - Cached - Similar pages

IEEE **Computer** Society
...September 2000 Vote for **Computer** Society officers! (members only) Members...
...nonmembers alike: Explore the **Computer** Society's magazines, journals,...
Description: The IEEE **Computer** Society is the original and largest worldwide organization serving as the leading...
Category: Science > Institutions > Associations > Computer Science
www.computer.org/ - 27k - Cached - Similar pages

Computer und Internet - Excite Deutschland
...hier! Ihr Standort: Homepage > **Computer** Internet **Computer**...
...Inhalt: **Computer** & Internet **COMPUTER** INTERNET Download...
excite.de.netscape.com/computer/ - 28k - Cached - Similar pages

Ranking:

- return best pages first
- term- vs. link- vs. click-based approaches
- machine-learned ranking to combine signals
- transformer-based ranking



Main Challenges for Large Search Engines:

- coverage
- good ranking
- freshness
- user load
- manipulation

Main Challenges for Large Search Engines:

- coverage (need to cover large part of the web)
  - → *need to crawl and store massive data sets*
- good ranking (cases of broad and narrow queries)
  - → *smart information retrieval techniques*
- freshness (need to update content)
  - → *frequent recrawling and reindexing of content*
- user load (> 50000 queries/sec - Google)
  - → *many queries on massive data*
- manipulation (sites want to be listed first)
  - → *naïve techniques will be exploited quickly*

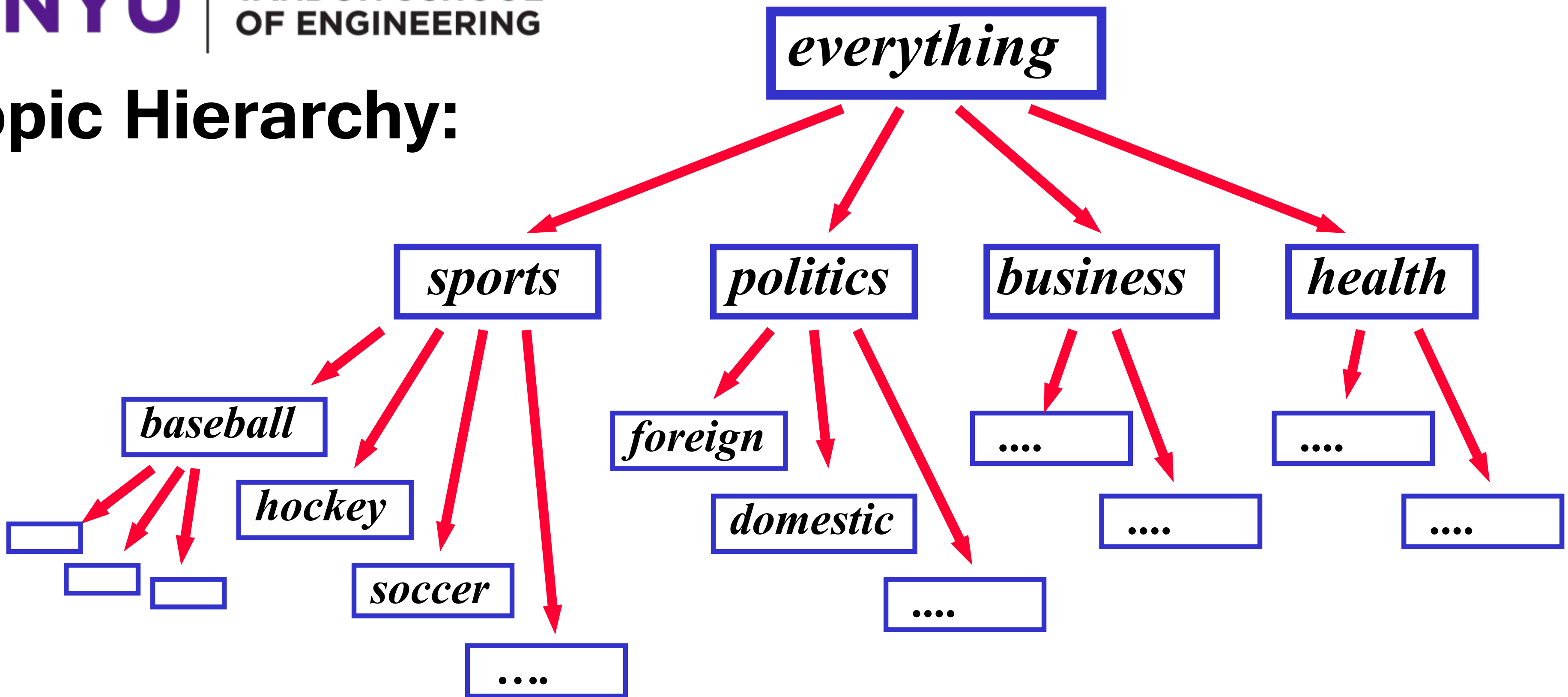
Main Challenges for Large Search Engines:

- coverage (need to cover large part of the web)
  → *need to crawl and store massive data sets*
- good ranking (cases of broad and narrow queries)
  → *smart information retrieval techniques*
- freshness (need to update content)
  → *frequent recrawling and reindexing of content*
- user load (> 50000 queries/sec - Google)
  → *many queries on massive data*
- manipulation (sites want to be listed first)
  → *naïve techniques will be exploited quickly*
- plus monetization, personalization, localization

Web Directories: *(Yahoo in mid 1990s)*

- based on categories or topics
- organize web pages in hierarchy of topics
- not sustainable in the end

Topic Hierarchy:



Challenges:

- designing topic hierarchy (in a fair way)
- automatic classification: *“what is this page about?”*
- Yahoo! and Open Directory were mostly human-based
- Compare to library classification (Dewey, LoC)

NYU | TANDON SCHOOL OF ENGINEERING

Specialized Search Engines: *(news, legal, medical)*

- be the best on one particular topic
- use domain-specific knowledge and structure
- limited resources → do not crawl the entire web!
- focused crawling, or meta search, or other data sources

Meta Search Engines:

- also related to "federated search"
- uses other search engines to answer queries
- e.g., ask the right specialized search engine
- or combine/rerank results from several engines
- or rewrite query using domain knowledge and send to engine
- not clear how well this works

E-Commerce Search: *(amazon, ebay, google shopping, …)*

- also called "product search"
- queries against a more or less structured "product catalog"
- semi-structured search: keywords plus product attributes
- e.g.: "red microwave oven" AND price < $200
- product attributes may depend on type of product
- e.g.: weight of laptop, power (watt) of microwave, etc.
- organize results by categories, or sort by attributes
- sometimes attributes extracted from text
- many verticals: travel, real estate, maybe resumes?

Sponsored Search: Computational Advertising

- placing the “best” ads next to search results
- or placing ads on web pages when they are visited
- sort of a search problem, but with different objective ($)
- financial foundation of “free” search engines
- use ad and page features and past user actions to pick ads
- major impediment to privacy on the web
- sponsored search runs more queries than organic search
- very complex system, could be an entire grad course

Recommender Systems: *(netflix, amazon, spotify, ...)*

- recommends items (e.g., movies) a user might like
- based on past likes, or likes of similar users
- the right answer depends on who is asking
- very close to search, but a little different: "recsys" community

Question Answering:

- can we understand and directly answer user questions
- "what is Abraham Lincoln's birthday?" or just "lincoln birthday"
- "how to fix a clogged toilet?", or "what is the meaning of life?"
- needs NLP and deeper text analysis
- recent progress based on LLMs

NYU | TANDON SCHOOL OF ENGINEERING

Multimedia search: *(google image/video search, music search)*

- based on keywords, or finding similar images or music
- useful for "instance retrieval" in AR apps
  - identify object in image by searching a labeled set of images
- extraction and indexing of multimedia features
- image or signal processing techniques: ICASSP, CVPR

Privacy-Preserving Search *(duckduckgo, neeva)*

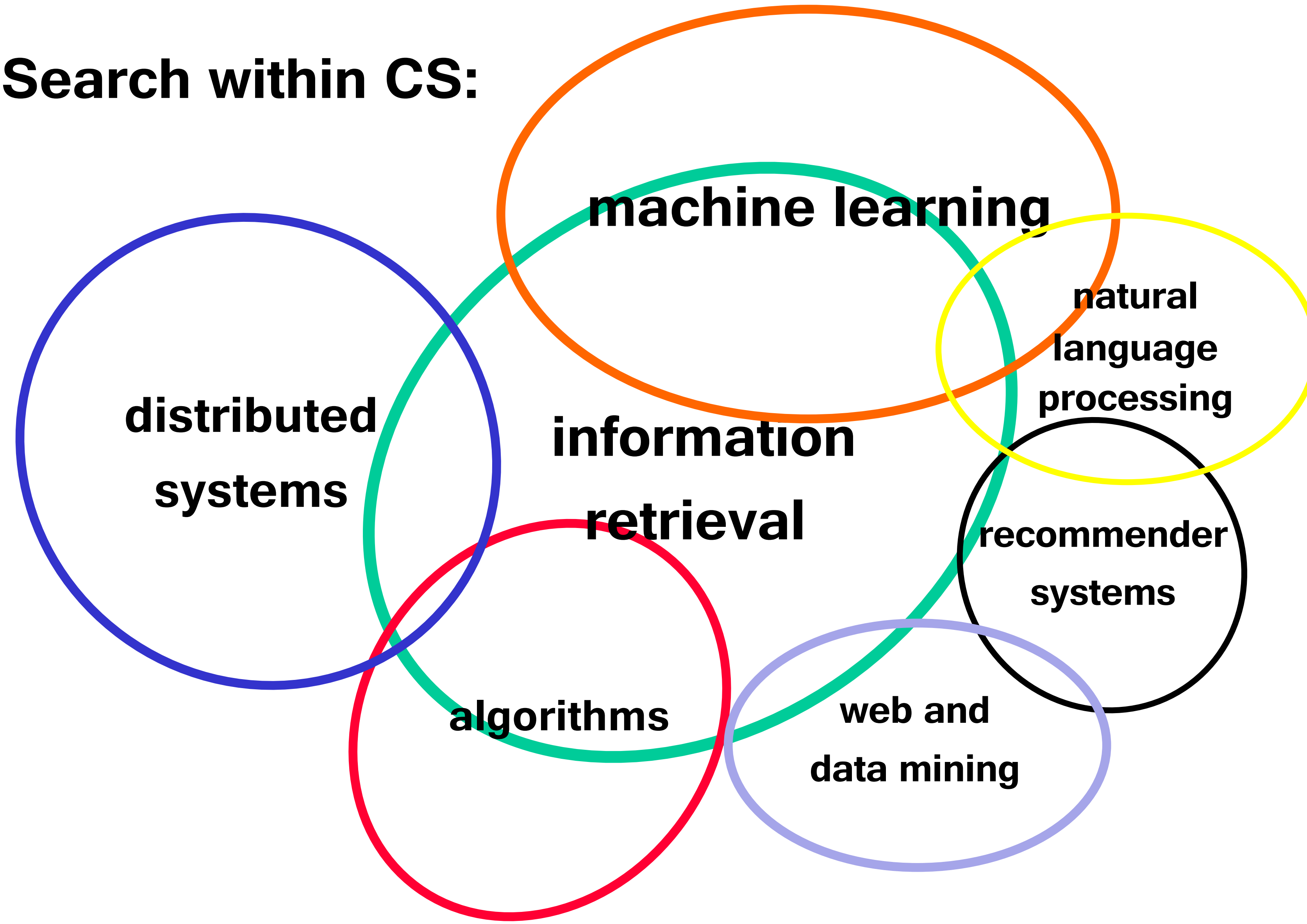
- how to search without revealing all your secrets
- partly a business problem, but also hard challenges
- private information retrieval (PIR) as a theory problem

NYU | TANDON SCHOOL OF ENGINEERING

Summary: Search Tools

- there are many search scenarios beyond Google etc.
- every large site and community needs search
- twitter, facebook, amazon, ebay, spotify
- techniques may vary, but common principles
- we focus on Google etc. but also consider other cases

Web Search within CS:



NYU TANDON SCHOOL OF ENGINEERING

Lecture 1: Introduction

I. The Web and Web Search:

- how the web works
- search tools

II. Introduction to Information Retrieval

- origins of IR
- basic setups and techniques

III. Main Components of a Search Engine

- basic search engine architecture
- web crawling
- indexing
- querying and ranking

Information Retrieval (IR):

"IR is concerned with the representation, storage, organization of, and access to information items"

- **focus on automatic processing (indexing, clustering, search) of unstructured data (text, images, audio, ...)**
- **subfield of Computer Science, but with roots in Library Science, Information Science, and Linguistics**
- ***In this course*, we mainly focus on text data**
- **Applications:**
  - **searching a library catalog**
  - **navigating a collection of medical, news, technical, or legal articles**
  - **since 1995: web search engines**

Historical Perspective:

- WW2 era computers built for number crunching:
  ballistics computations, code breaking
- since earliest days, also used to "organize information"
  - Memex (Vannevar Bush, 1945)
- today, this is the main application!
  - store and organize data and documents
  - model organizations and work processes
- ~~Computer~~ Organizer
- ... however, no clean separation

Structured vs. Unstructured Data:

- IR: lesser known cousin of the field of *Databases*
- Databases: focus on structured data

customer-id	*customer-name*	*customer-street*	*customer-city*
192-83-7465	Johnson	12 Alma St.	Palo Alto
019-28-3746	Smith	4 North St.	Rye
677-89-9011	Hayes	3 Main St.	Harrison
182-73-6091	Turner	123 Putnam Ave.	Stamford
321-12-3123	Jones	100 Main St.	Harrison
336-66-9999	Lindsay	175 Park Ave.	Pittsfield
019-28-3746	Smith	72 North St.	Rye

(a) The *customer* table

account-number	*balance*
A-101	500
A-215	700
A-102	400
A-305	350
A-201	900
A-217	750
A-222	700

(b) The *account* table

- IR: unstructured data: "documents"
  - scientific articles, novels, poems, jokes, web pages, images
- unstructured data is not really unstructured
- Information retrieval vs. data retrieval?
- IR focused on human users and their needs
- DB optimized to serve as building block for higher-level apps

NYU | TANDON SCHOOL OF ENGINEERING

Evaluation in IR: Precision versus Recall

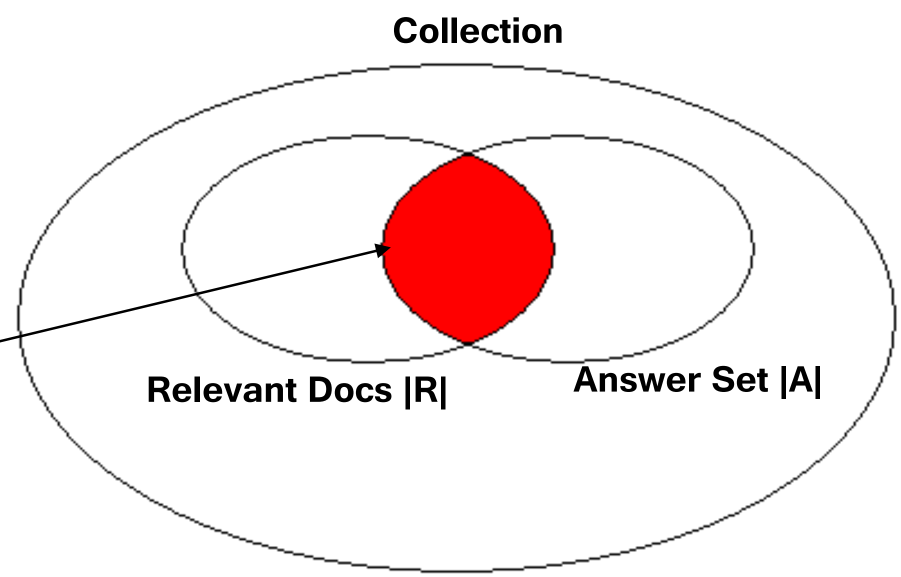
Search Problem: *"Given 1 million text documents, find all documents that are about kangaroos"*

Recall: the fraction of the relevant documents that are retrieved:

$$\text{recall} = \frac{|R_a|}{|R|}$$

Precision: the fraction of retrieved documents (A) that are relevant:

$$\text{precision} = \frac{|R_a|}{|A|}$$



Relevant documents in answer set $|R_a|$

NYU | TANDON SCHOOL OF ENGINEERING

Example: Precision versus Recall

- Suppose there are 160 documents about kangaroos
- Our search tools return 100 documents
- Of these 100 documents, 80 are about kangaroos

Example: Precision versus Recall

- Suppose there are 160 documents about kangaroos
- Our search tools return 100 documents
- Of these 100 documents, 80 are about kangaroos

- Precision is 80/100 = 0.8
- Recall is 80/160 = 0.5

NYU | TANDON SCHOOL OF ENGINEERING

Example: Precision versus Recall

- Suppose there are 160 documents about kangaroos
- Our search tools return 100 documents
- Of these 100 documents, 80 are about kangaroos

- Precision is 80/100 = 0.8
- Recall is 80/160 = 0.5

- Also used for many ML problems

NYU TANDON SCHOOL OF ENGINEERING

How do you know that a document is relevant to a topic or query?

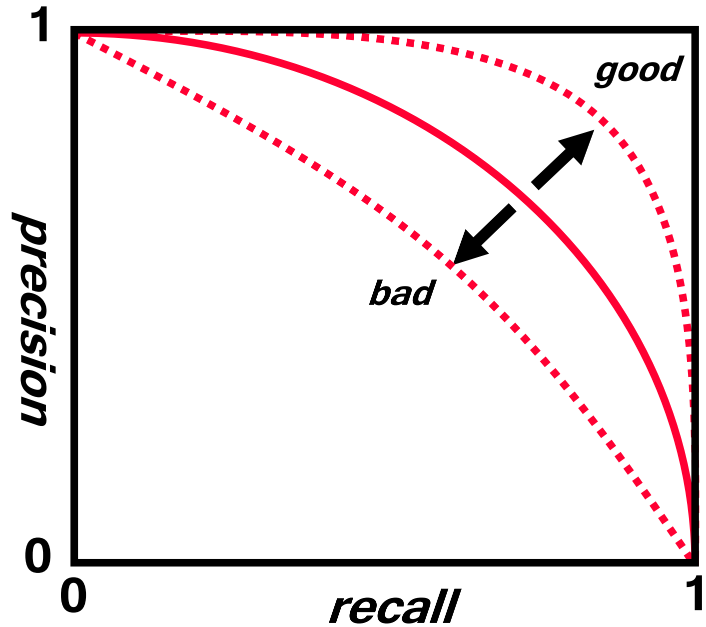
How do you know that a document is relevant to a topic or query?

You ask somebody!

An expert, a student volunteer

People getting paid to label things

- Problem: *"return documents relevant to a particular query/topic"*
- Extreme cases:
  - if system returns no result: precision 100% but recall 0%
  - if system returns all documents as result: recall 100% but precision ~0%
- Maybe in the middle:
  - 8 relevant results in collection
  - 6 relevant results in 10 returned results
  - recall = 0.75, precision = 0.6
- fundamental trade-off



- Real IR systems use more complex evaluation measures

Evaluation in IR: More Realistic Measures

- Often only interested in top-ranked results (recall not that useful)
- Popular measure: precision@k (p@k)
- For example, p@10: what fraction of top-10 results is relevant?
- p@10 = 0.59 → over many queries, about 59% of top 10 are relevant

- But we should also take degree of relevance and position into account
- We want *highly relevant* results at the very top

- Other measures: map, ndcg, rbp, and many others
- Ongoing area of research

Typical Operations in IR Systems

- **Indexing:** create a full-text or keyword index
- **Querying/ranking:** find documents most relevant to a user query
- **Clustering:** group documents in sets of similar ones
- **Categorization:** assign documents to given set of categories
- **Citation Analysis:** find frequently cited or influential papers

Problems more typically addressed by NLP

- **Summarization:** automatically create a summary
- **Machine translation:** translate text between languages
- **Info extraction/tagging:** identify names of people, organizations

IR different from NLP and from String Processing

IR versus NLP: (Natural Language Processing, or Computational Linguistics)

- IR usually does not analyze grammar, local/sentence structure *(document as a set or bag or words - mostly)*
- NLP analyzes sentence structure *(shallow/deep parsing)*
- IR: simple statistical methods, good for search & categorization
- NLP: good for automatic translation, summarization, extraction
- IR is largely language-independent
- NLP uses more knowledge about the language *(grammar, thesaurus)*
- NLP: rooted in linguistics, grammar vs. statistical NLP
- Web Search: NLP has proven useful in many cases
  - extraction/tagging: finding references to people, orgs, places
  - analyzing and understanding user queries
  - often more important to understand queries than docs (user intent)

NYU TANDON SCHOOL OF ENGINEERING

IR versus Recommender Systems

IR versus Recommender Systems

- IR usually assumes some commonality of user interests
- RecSys assumes users have different preferences

- In IR, the focus is on answering search queries
- RecSys do not need queries (Spotify: “find some music I like”)

- However, IR and RecSys are converging as IR systems need to model personal preferences, and RecSys need to support queries on large data stets

Early Historical Development of IS and IR

(IS = Information Science)

- **Babylonians, Greeks, Romans, etc.**
- **Indexing and creation of concordances**
  - algorithms for (full-)text indexing
  - e.g., Dobson's *Byron* concordance (1940-65, "last of the handmade concordances")
- ***Library of Congress* and *Dewey* Library Classifications**
- **Documentalism: movement in early 1900s to organize all the information in the world in a structured form**
- **Microfilm rapid selectors and the Memex**
- **Bibliometric and Informetric distributions:**
  - Bradford, Lotka, Zipf, Pareto, Yule (1920s-40s)
- **Citation Analysis and Social Network Analysis (1950s)**

Rapid Microfilm Selectors:

- microfilm for storage of large amounts of data
- storage density in MB per square inch (1925)
- but how can we search such massive data?
- idea:
  - add index terms to microfilm boundaries
  - build rapid selection machines that can scan index
- Rapid Microfilm Selectors:
  - use light and photo cells to find matches in index terms
  - scan hundreds of index cards per second

Source: M. Buckland, UC Berkeley
See http://www.sims.berkeley.edu/~buckland/goldbush.html

Rapid Microfilm Selectors:

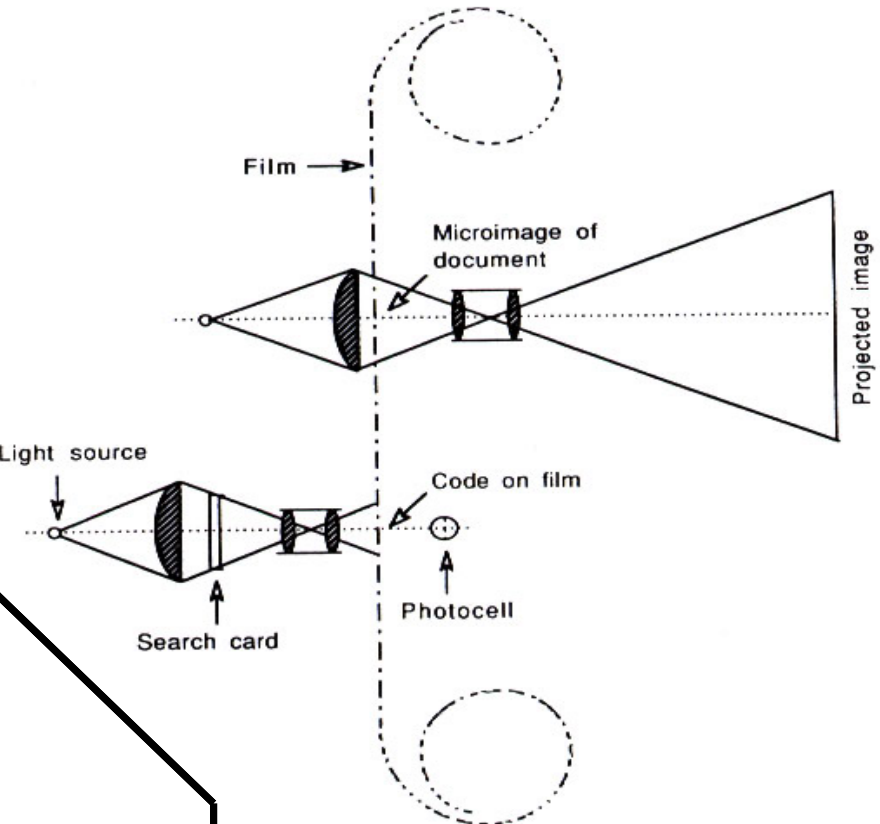
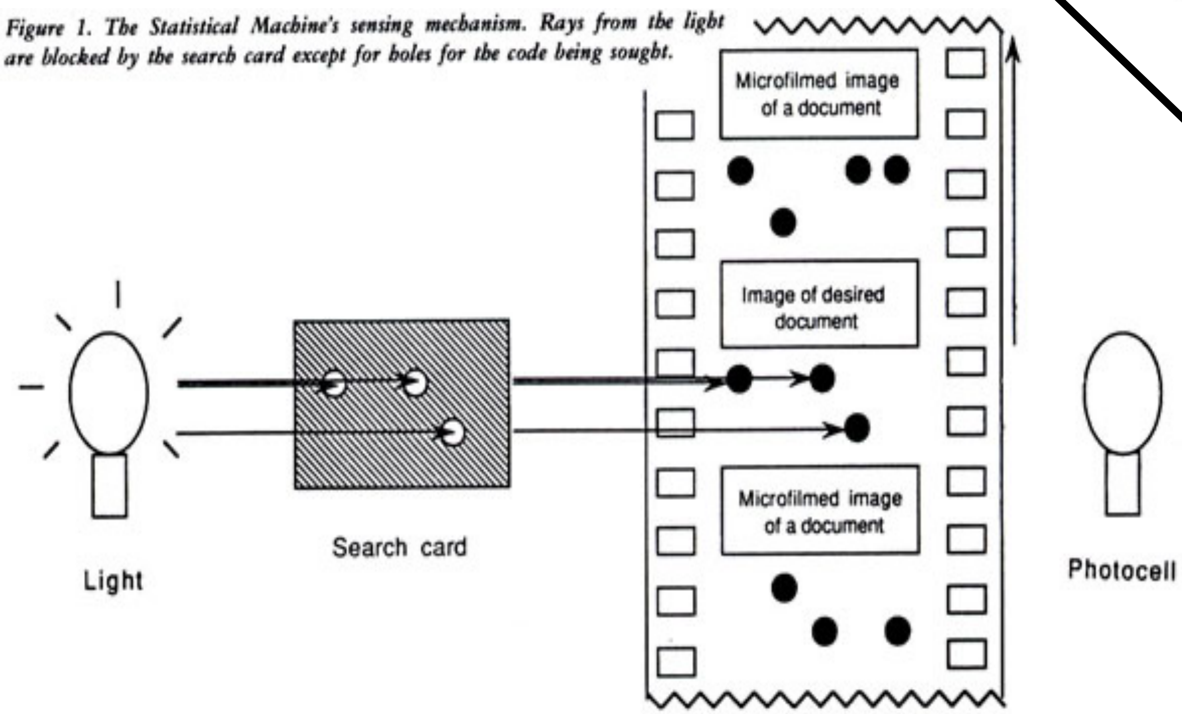
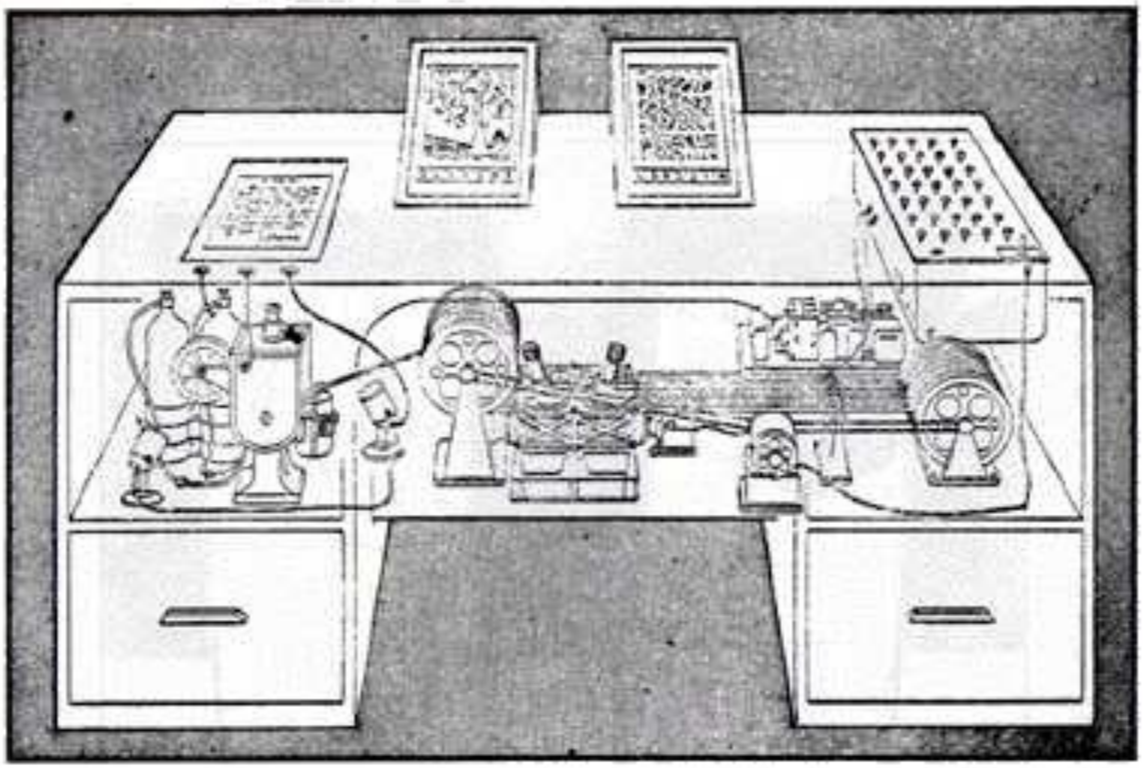


Figure 1. The Statistical Machine's sensing mechanism. Rays from the light are blocked by the search card except for holes for the code being sought.



Memex: Vannevar Bush (1890-1974)

- "As We May Think", *Atlantic Monthly*, 1945 ***(mostly written in1939)***



Memex in the form of a desk would instantly bring files and material on any subject to the operator's fingertips. Slanting translucent viewing screens magnify supermicrofilm filed by code numbers. At left is a mechanism which automatically photographs longhand notes, pictures and letters, then files them in the desk for future reference (*LIFE* 19(11), p. 123).

Memex:

- device for storing and retrieving information
- storage based on microfilm
- users can store all their experiences and knowledge and retrieve information later
- *trails* allow users to link different pieces of information
- Basically, Memex proposed the idea of hyperlinks!

Zipf and Bibliometric/Informetric Distributions

- **distributions observed in many fields of science**
- **some things occur much more frequently than others**

  "some authors are much more often cited than others"

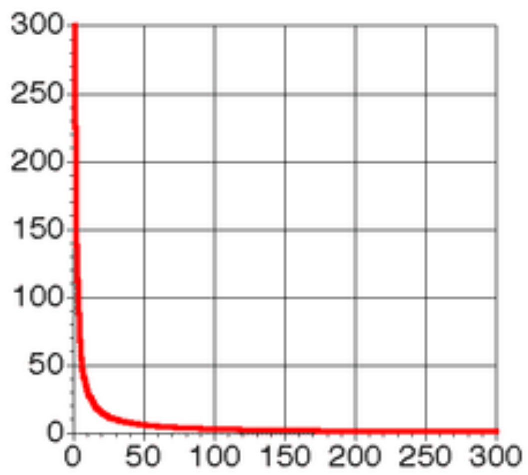
  "some people have much more money than others"

  "some web pages have more in-links than others"

  "some books or songs are more popular than others"

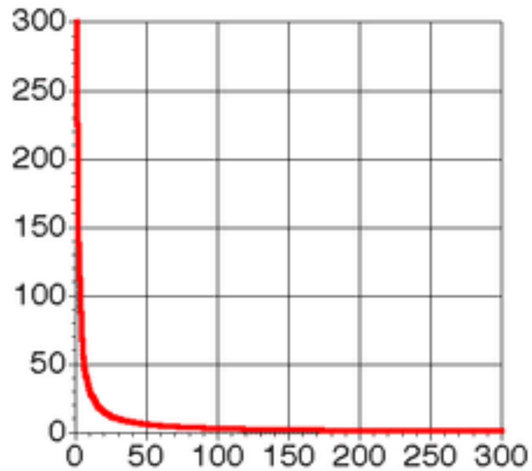
  "some words are much more often used than others"

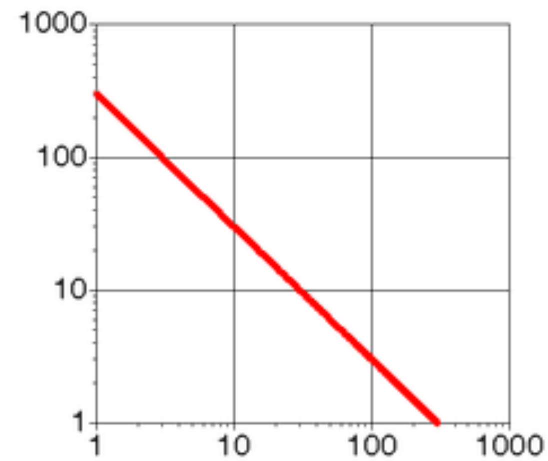
- **often follows a particular class of distributions:** $f(i) \sim i^{-z}$



Zipf and Bibliometric/Informetric Distributions (ctd.)

$f(i) \sim i^{-z}$





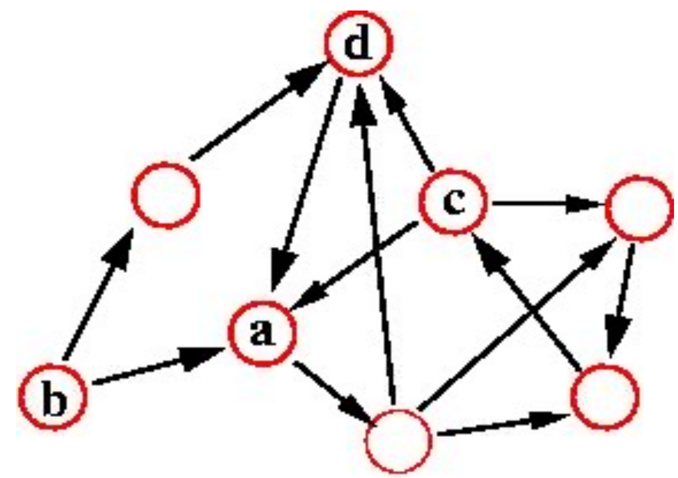
- Zipf distributions, power laws, Pareto distribution, ... (not always the same)
- larger *z* means larger skew
- heavy-tailed: *"some may have a lot more, but most stuff is owned by the many"*
- non heavy-tailed: *"some have a lot more, and own almost everything"*
- similar observations in economics, social sciences, biology, etc.
- e.g., relevance to music distribution - from broadcast to personalized

Citation and Social Network Analysis

- "who has written the most influential papers in physics?"
- "who is the most influential person in a social network?"
- maybe the person who knows the largest number of people?
- or someone who knows a few influential people very well?

→ graph-theoretic approaches to analyzing social networks and citation graphs

(Katz 1953, Garfield 1967)



s(a) ~ s(b) + s(c) + s(d) ?

- national security applications:
  - funding, communication, coordination
  - telephone call graphs, email graphs
- social networks (facebook, twitter, wechat)
- sociology

Note: degrees in networks often have Zipf (power law) properties

Historical Development of IR (after 1960)

- early work by H.-P. Luhn (KWIC index, SDI, abstraction)
- hypertext (Nelson, Engelbart, 1960s)
  - links between different documents and sections
  - Xanadu system, hypertext community (HT conferences)
- Cranfield experiments (Cleverdon, 1960s)
- vector space model and ranking methods
  - Salton et al (Cornell), 1960s
  - cosine measure, SMART system
  - probabilistic relevance models (1970s)
- automatic text classification using ML (1960s to now)
- world wide web, Berners-Lee, 1991
  - earlier: gopher, archie, WAIS
  - 1994: Mosaic browser, breakthrough in size of the web
  - 1994/1995: first generation of crawler-based commercial search engines

Lecture 1: Introduction

I. The Web and Web Search:

- how the web works
- search tools

II. Introduction to Information Retrieval

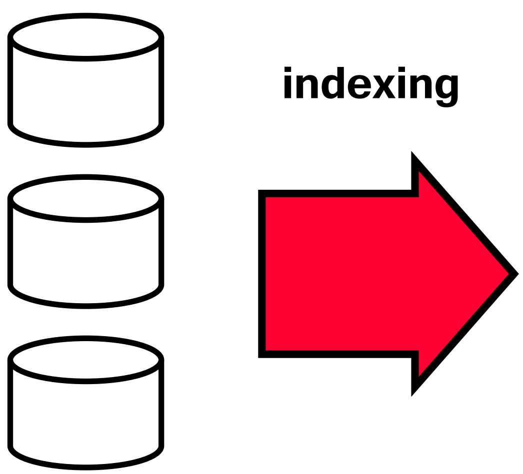
- origins of IR
- basic setups and techniques

III. Main Components of a Search Engine

- basic search engine architecture
- web crawling
- indexing
- querying and ranking

Text Index:

- a data structure that for supporting IR queries
- most popular form: *inverted index structure*
- like the index of a book



disks with documents

aalborg	3452, 11437, ...
.	
.	
.	
.	
.	
arm	4, 19, 29, 98, 143, ...
armada	145, 457, 789, ...
armadillo	678, 2134, 3970, ...
armani	90, 256, 372, 511, ...
.	
.	
.	
.	
.	
zebra	602, 1189, 3209, ...

inverted index

Boolean Querying:

Boolean queries: cafes in Brooklyn

cafe AND Brooklyn

aalborg	3452, 11437, ...
.	
.	
.	
.	
arm	4, 19, 29, 98, 143, ...
armada	145, 457, 789, ...
armadillo	678, 2134, 3970, ...
armani	90, 256, 372, 511, ...
.	
.	
.	
.	
zebra	602, 1189, 3209, ...

look up

Boolean Querying:

Boolean queries: cafes in Brooklyn

cafe OR Brooklyn

aalborg	3452, 11437, ...
:	
arm	4, 19, 29, 98, 143, ...
armada	145, 457, 789, ...
armadillo	678, 2134, 3970, ...
armani	90, 256, 372, 511, ...
:	
zebra	602, 1189, 3209, ...

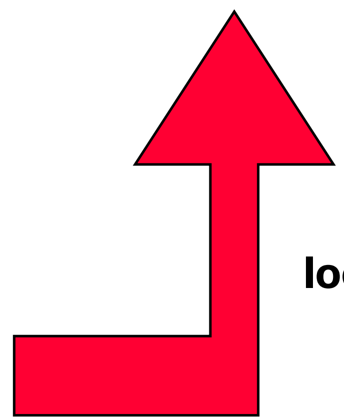
look up

Boolean Querying:

Boolean queries: cafes in Brooklyn

(cafe OR restaurant) AND Brooklyn

aalborg	3452, 11437, ...
.	
.	
.	
.	
arm	4, 19, 29, 98, 143, ...
armada	145, 457, 789, ...
armadillo	678, 2134, 3970, ...
armani	90, 256, 372, 511, ...
.	
.	
.	
.	
zebra	602, 1189, 3209, ...



look up

NYU TANDON SCHOOL OF ENGINEERING

Boolean Querying:

Boolean queries: cafes in Brooklyn

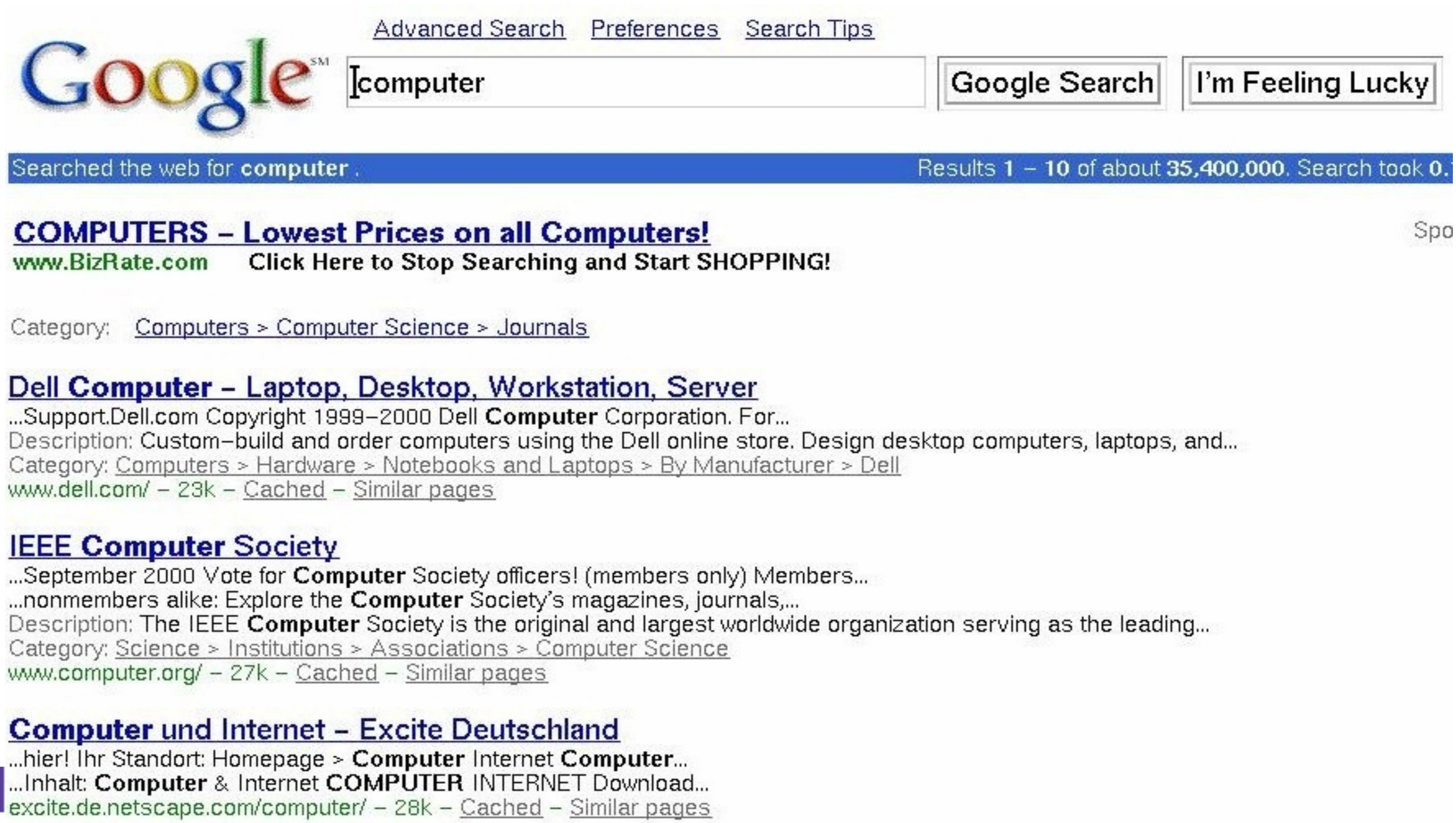
(cafe OR restaurant) AND Brooklyn

→ **unions/intersections of sorted lists**

aalborg	3452, 11437, ...
.	
.	
arm	4, 19, 29, 98, 143, ...
armada	145, 457, 789, ...
armadillo	678, 2134, 3970, ...
armani	90, 256, 372, 511, ...
.	
.	
zebra	602, 1189, 3209, ...

look up

Ranking: "return best documents first"



NYU | TANDON SCHOOL OF ENGINEERING

Ranking: "return best documents first"

- what does that mean?

Ranking: "return best documents first"

- what does that mean?
- who decides what is "best"?

Ranking: “return best documents first”

- what does that mean?
- who decides what is “best”?
- Assumption: the user decides what is best

Ranking: "return best documents first"

- what does that mean?
- who decides what is "best"?
- Assumption: the user decides what is best

  "make the user happy"

- Problem #1: we do not know what the user wants
- Problem #2: does the user know what she wants?
- Problem #3: relevance versus importance versus quality/veracity
- Problem #4: system goals, fairness, and equity

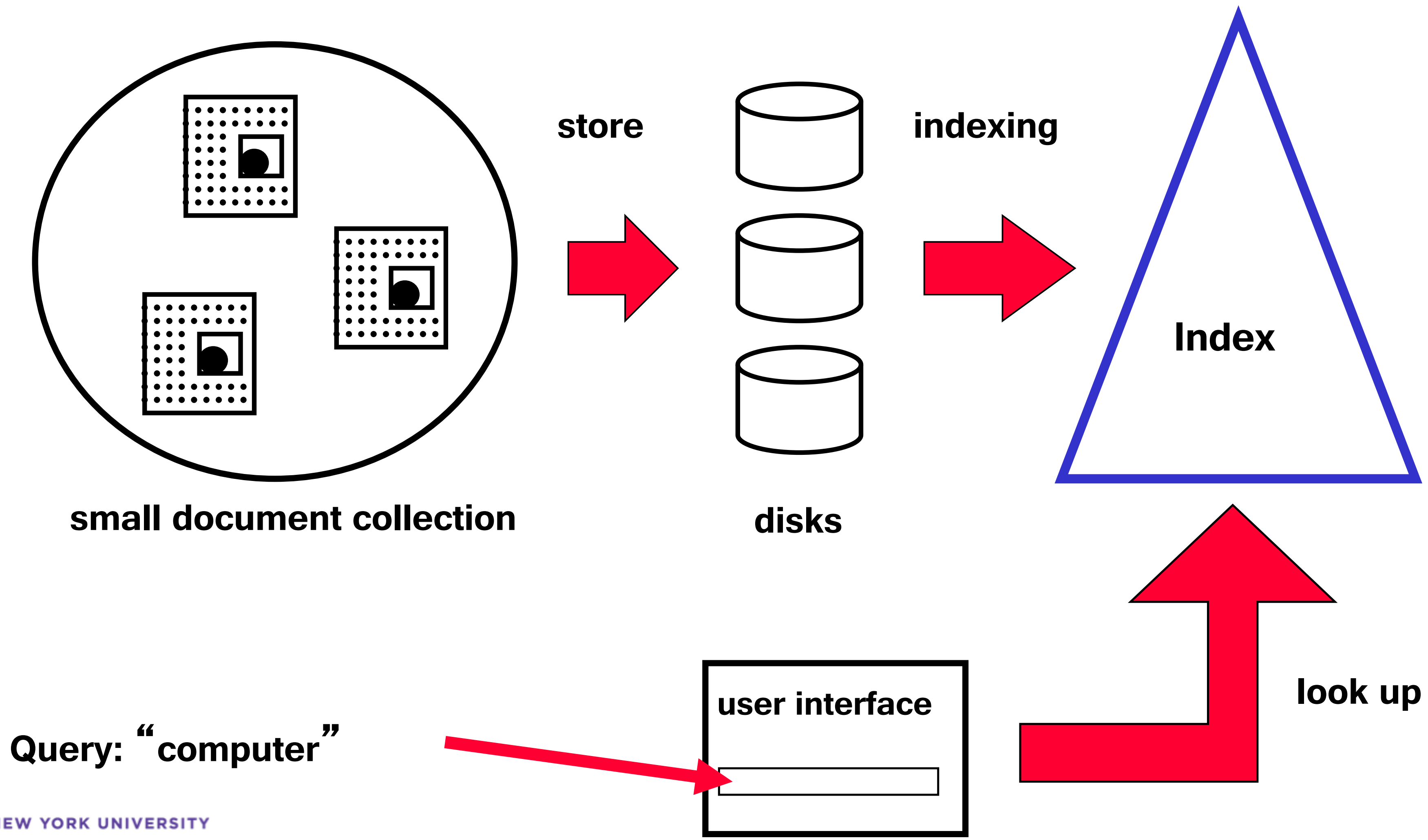
Ranking: "return best documents first"

- but basically, we need human judgments to evaluate

Ranking: "return best documents first"

- but basically, we need human judgments to evaluate
- traditionally, IR researchers collect feedback from paid or unpaid volunteers
- Cranfield methodology for evaluating search tools (since 1960s)
- today, search engines pay millions per quarter to get feedback

Basic structure of a simple traditional IR system



Simple Traditional IR System (before the web)

- constructs and maintains inverted index on documents
- may supports Boolean or ranked queries
- may use automatic or manual classification
- may support other clustering techniques
- may support advanced browsing operations

- "searching vs. browsing"
- often small well-structured collections *(news, medical articles)*
- queries with many keywords *(up to hundreds)*

Web Search vs. Traditional IR Systems:

- data acquisition important *(crawling the web)*
- collections are much larger *(3 billion pages = 50 TB)*
- documents are of very mixed quality and types
- queries are VERY short *(less than 3 words on average)*
- traditional statistical techniques do not work as well
- we have additional features we can use:
  - hyperlink structure
  - usage data / logs (of course, privacy issues)
- on the other hand, there is search engine manipulation!

Foundations: Vector-Space Model in IR

- each document *D* represented as set of words
- a query *Q* is also just a set of words
- let *L* be the set of all words in the collection, $|L| = m$
- *D* and *Q* correspond to *m*-dimensional vectors
  - if word does not occur in D resp. Q, the corresponding element is set to 0
  - otherwise, element is positive
- score of *D* with respect to query *Q* is $D * Q$
- return documents with highest *k* scores

Vector-Space Model (ctd.)

- **Example: put a 1 into vector for each word**

L = {a, alice, bob, book, like, reads}, m = 6

doc1: "Bob reads a book"	*D1* = (1, 0, 1, 1, 0, 1)
doc2: "Alice likes Bob"	*D2* = (0, 1, 1, 0, 1, 0)
doc3: "book"	*D3* = (0, 0, 0, 1, 0, 0)

Vector-Space Model (ctd.)

- **Example: put a 1 into vector for each word**

L = {a, alice, bob, book, like, reads}, m = 6

doc1: "Bob reads a book" — *D1* = (1, 0, 1, 1, 0, 1)

doc2: "Alice likes Bob" — *D2* = (0, 1, 1, 0, 1, 0)

doc3: "book" — *D3* = (0, 0, 0, 1, 0, 0)

document matrix

Vector-Space Model (ctd.)

- Example: put a 1 into vector for each word

L = {a, alice, bob, book, like, reads}, m = 6

doc1: "Bob reads a book"	*D1* = (1, 0, 1, 1, 0, 1)
doc2: "Alice likes Bob"	*D2* = (0, 1, 1, 0, 1, 0)
doc3: "book"	*D3* = (0, 0, 0, 1, 0, 0)
query: "bob, book"	*Q* = (0, 0, 1, 1, 0, 0)

Vector-Space Model (ctd.)

- **Example: put a 1 into vector for each word**

L = {a, alice, bob, book, like, reads}, m = 6

doc1: "Bob reads a book" — *D1* = (1, 0, 1, 1, 0, 1)

doc2: "Alice likes Bob" — *D2* = (0, 1, 1, 0, 1, 0)

doc3: "book" — *D3* = (0, 0, 0, 1, 0, 0)

query: "bob, book" — *Q* = (0, 0, 1, 1, 0, 0)

query vector

*D1*Q = 2*, *D2 * Q = 1*, *D3 * Q = 1*

Vector-Space Model (ctd.)

- Example: put a 1 into vector for each word

L = {a, alice, bob, book, like, reads}, m = 6

doc1: "Bob reads a book"	*D1* = (1, 0, 1, 1, 0, 1)
doc2: "Alice likes Bob"	*D2* = (0, 1, 1, 0, 1, 0)
doc3: "book"	*D3* = (0, 0, 0, 1, 0, 0)
query: "bob, book"	*Q* = (0, 0, 1, 1, 0, 0)

*D1*Q = 2*, *D2 * Q = 1*, *D3 * Q = 1*

- very primitive ranking function: "how many words in common?"
- smarter functions: assign appropriate weights to doc vectors
- vector-matrix multiplication to score are documents

Vector-Space Model (ctd.)

- higher score for more occurrences of a word
- higher score for rare words
- lower score for long documents

NYU | TANDON SCHOOL OF ENGINEERING

Vector-Space Model (ctd.)

- higher score for more occurrences of a word
- higher score for rare words
- lower score for long documents
- example: "*cosine measure*" (and many others)

$$F(d, t_0, \ldots, t_{m-1}) = \sum_{i=0}^{m-1} \frac{w(q, t_i) \cdot w(d, t_i)}{\sqrt{|d|}},$$
$$w(q, t) = \ln(1 + N/f_t), \text{and}$$
$$w(d, t) = 1 + \ln f_{d,t},$$

- $f_{d,t}$ = number of occurrences of term *t* in document *d*
- f_t = total number of occurrences of *t* in the collection
- *N* = total number of documents

$$F(d, t_0, \ldots, t_{m-1}) = \sum_{i=0}^{m-1} \frac{w(q, t_i) \cdot w(d, t_i)}{\sqrt{|d|}},$$

$$w(q, t) = \ln(1 + N / f_t), \text{ and}$$

$$w(d, t) = 1 + \ln f_{d,t},$$

- $f_{d,t}$ = number of occurrences of term *t* in document *d*
- f_t = total number of occurrences of *t* in the collection of *N* documents

Previous Example:

L = {a, alice, bob, book, like, reads}, m = 6

doc1: "Bob reads a book" *D1* = (1, 0, 1, 1, 0, 1)

doc2: "Alice likes Bob" *D2* = (0, 1, 1, 0, 1, 0)

doc3: "book" *D3* = (0, 0, 0, 1, 0, 0)

query: "bob, book" *Q* = (0, 0, 1, 1, 0, 0)

*D1*Q = 2, D2 * Q = 1, D3 * Q = 1*

Cosine:

Note: *N* = 3, and $f_{d,t}$ is either 0 or 1 for all *d* and *t*, since no doc contains a word twice

doc1: "Bob reads a book" *D1* = ($\frac{1}{2}$, 0, $\frac{1}{2}$, $\frac{1}{2}$, 0, $\frac{1}{2}$)

doc2: "Alice likes Bob" *D2* = (0, $\frac{1}{\sqrt{3}}$, $\frac{1}{\sqrt{3}}$, 0, $\frac{1}{\sqrt{3}}$, 0)

doc3: "book" *D3* = (0, 0, 0, 1, 0, 0)

query: "bob, book" *Q* = (0, 0, *ln*(2.5), *ln*(2.5), 0, 0)

*D1*Q* = $\frac{1}{2}$ *ln*(2.5) + $\frac{1}{2}$ *ln*(2.5) *D2 * Q* = $\frac{1}{\sqrt{3}}$ *ln*(2.5) *D3 * Q = ln*(2.5)

- **another popular (usually better) function: BM25**

$$BM25(q,d) = \sum_{t \in q} \log\left(\frac{N - f_t + 0.5}{f_t + 0.5}\right) \times \frac{(k_1 + 1) f_{d,t}}{K + f_{d,t}}$$

$$K = k_1 \times \left((1-b) + b \times \frac{|d|}{|d|_{avg}}\right)$$

- N: **total number of documents in the collection;**
- f_t: **number of documents that contain term** t;
- $f_{d,t}$: **frequency of term** t **in document** d;
- $|d|$: **length of document** d;
- $|d|_{avg}$: **the average length of documents in the collection;**
- k_1 **and** b: **constants, usually** $k_1 = 1.2$ **and** $b = 0.75$

NYU | TANDON SCHOOL OF ENGINEERING

Ranking and Retrieval Models

- vast amount of work on ranking in IR
- retrieval models: vector space, probabilistic, language, neural
- retrieval model: formal way of reasoning about ranking

Ranking and Retrieval Models

- vast amount of work on ranking in IR
- retrieval models: vector space, probabilistic, language, neural
- retrieval model: formal way of reasoning about ranking
- probabilistic retrieval model: (Robertson 1977)

  "rank documents according to their likelihood to be relevant to query"

Ranking and Retrieval Models

- vast amount of work on ranking in IR
- retrieval models: vector space, probabilistic, language, neural
- retrieval model: formal way of reasoning about ranking
- probabilistic retrieval model: (Robertson 1977)

  "rank documents according to their likelihood to be relevant to query"
- this allows us to derive ranking functions under certain assumptions
- e.g., Bayesian approaches, under assumptions about docs & queries

NYU | TANDON SCHOOL OF ENGINEERING

Ranking and Retrieval Models

- vast amount of work on ranking in IR
- retrieval models: vector space, probabilistic, language, neural
- retrieval model: formal way of reasoning about ranking
- probabilistic retrieval model: (Robertson 1977)

  "rank documents according to their likelihood to be relevant to query"
- this allows us to derive ranking functions under certain assumptions
- e.g., Bayesian approaches, under assumptions about docs & queries
- Language-modeling approach: document is relevant to a query if query and document are likely to be generated by same underlying random process

NYU | TANDON SCHOOL OF ENGINEERING

Ranking and Retrieval Models

- vast amount of work on ranking in IR
- retrieval models: vector space, probabilistic, language, neural
- retrieval model: formal way of reasoning about ranking
- *many* different ranking functions, based on different models

Ranking and Retrieval Models

- vast amount of work on ranking in IR
- retrieval models: vector space, probabilistic, language, neural
- retrieval model: formal way of reasoning about ranking
- *many* different ranking functions, based on different models
- additional factors in ranking (mainly for web):
  - higher weight if word in title, in large font, in bold face
  - search engines: higher score if word in URL, in anchortext
  - distance between terms in text (near, or far away?)
  - user feedback (clicks) or browsing behavior?
  - hyperlink structure

Ranking and Retrieval Models

- vast amount of work on ranking in IR
- retrieval models: vector space, probabilistic, language, neural
- retrieval model: formal way of reasoning about ranking
- *many* different ranking functions, based on different models
- additional factors in ranking (mainly for web):
  - higher weight if word in title, in large font, in bold face
  - search engines: higher score if word in URL, in anchortext
  - distance between terms in text (near, or far away?)
  - user feedback (clicks) or browsing behavior?
  - hyperlink structure
- baseline execution: "compute scores of all documents containing at least one query term, by scanning the inverted lists and ranking"

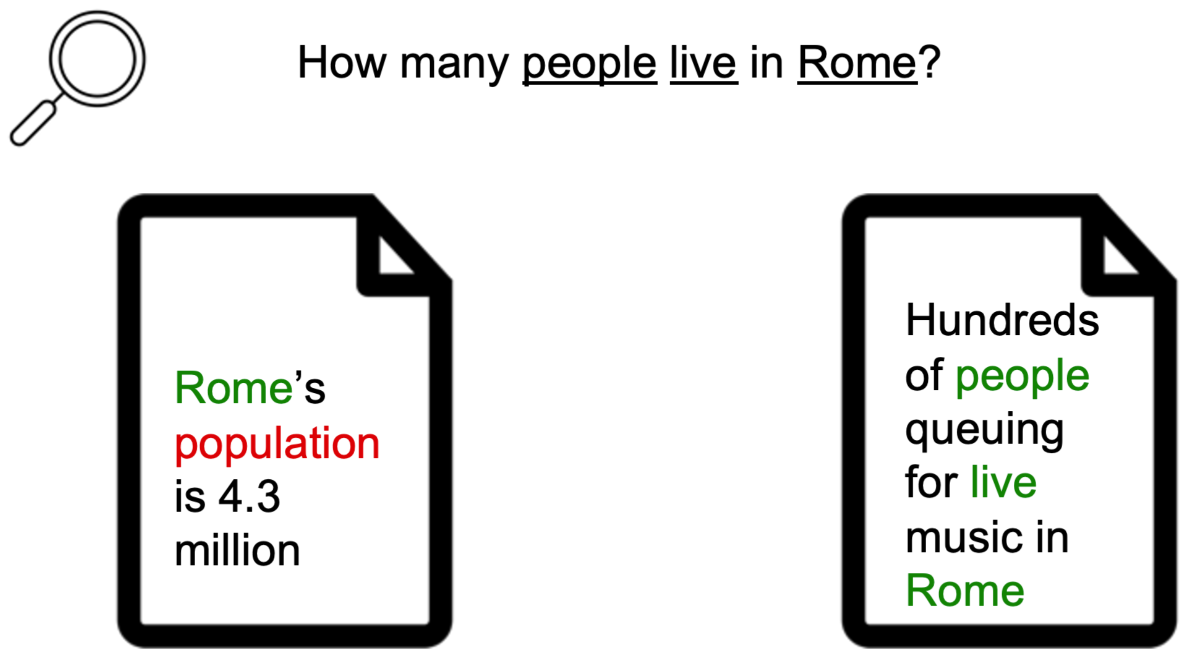
The Vocabulary Mismatch Problem

- many traditional ranking methods focus on term matching
- that is, do the query terms occur in the document?
- but such lexical similarity is not the same as semantic simularity

NYU TANDON SCHOOL OF ENGINEERING

The Vocabulary Mismatch Problem

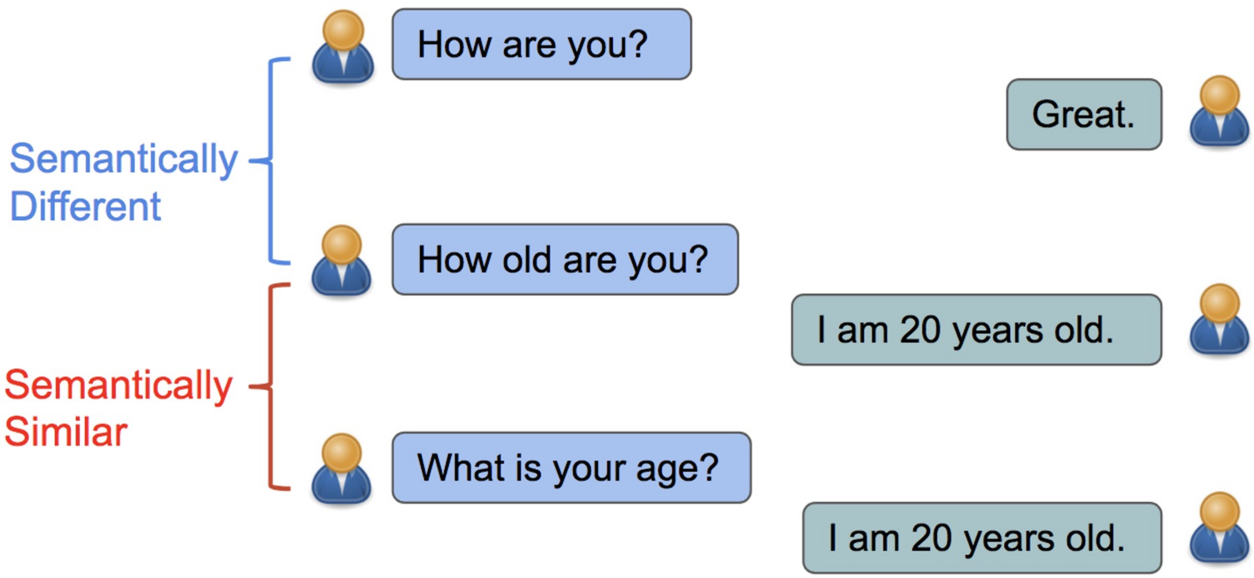
- many traditional ranking methods focus on term matching
- that is, do the query terms occur in the document?
- but such lexical similarity is not the same as semantic similarity



(Example due to Antonio Mallia)

The Vocabulary Mismatch Problem

- many traditional ranking methods focus on term matching
- that is, do the query terms occur in the document?
- but such lexical similarity is not the same as semantic simularity



https://ai.googleblog.com/2018/05/advances-in-semantic-textual-similarity.html

The Vocabulary Mismatch Problem

- due to differences in wording between user and document
- because there are many ways to ask same question
- synonyms or closely related terms: car/automobile,   boat/yacht
- or because users and document authors have different background

The Vocabulary Mismatch Problem

- due to differences in wording between user and document
- because there are many ways to ask same question
- synonyms or closely related terms: car/automobile, boat/yacht
- or because users and document authors have different background
- traditional approach: query expansion and relevance feedback

The Vocabulary Mismatch Problem

- due to differences in wording between user and document
- because there are many ways to ask same question
- synonyms or closely related terms: car/automobile,   boat/yacht
- or because users and document authors have different background
- traditional approach: query expansion and relevance feedback
- also, document expansion

The Vocabulary Mismatch Problem

- due to differences in wording between user and document
- because there are many ways to ask same question
- synonyms or closely related terms: car/automobile, boat/yacht
- or because users and document authors have different background
- traditional approach: query expansion and relevance feedback
- also, document expansion
- very recently: transformers and high-dimensional vector representations for modeling semantic similarity
- word2vec, BERT, nearest-neighbor approaches

Many Other Important Ideas in IR

- query logs and modeling user behavior
- relevance feedback and query expansion
- advanced issues in search system evaluation
- clustering and decomposition techniques (LSI, SVD)
- optimized index representation, access, and compression
- distributed search systems
- adversarial information retrieval
- learning to rank, and deep neural nets for ranking

NYU | TANDON SCHOOL OF ENGINEERING

Lecture 1: Introduction

I. The Web and Web Search:

- how the web works
- search tools

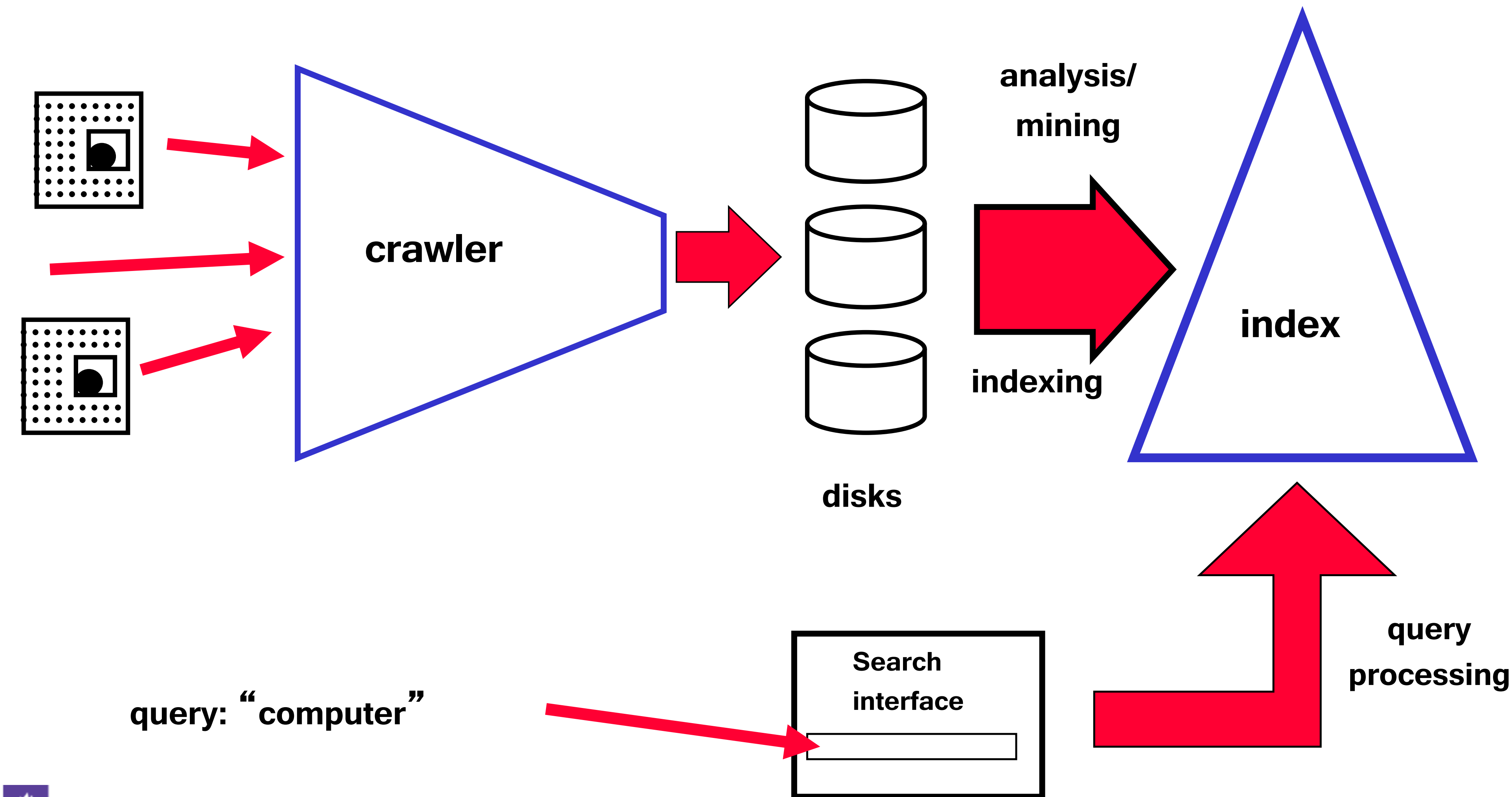
II. Introduction to Information Retrieval

- origins of IR
- basic setups and techniques

III. Main Components of a Search Engine

- basic search engine architecture
- web crawling
- indexing
- querying and ranking

Basic Structure of a Search Engine:

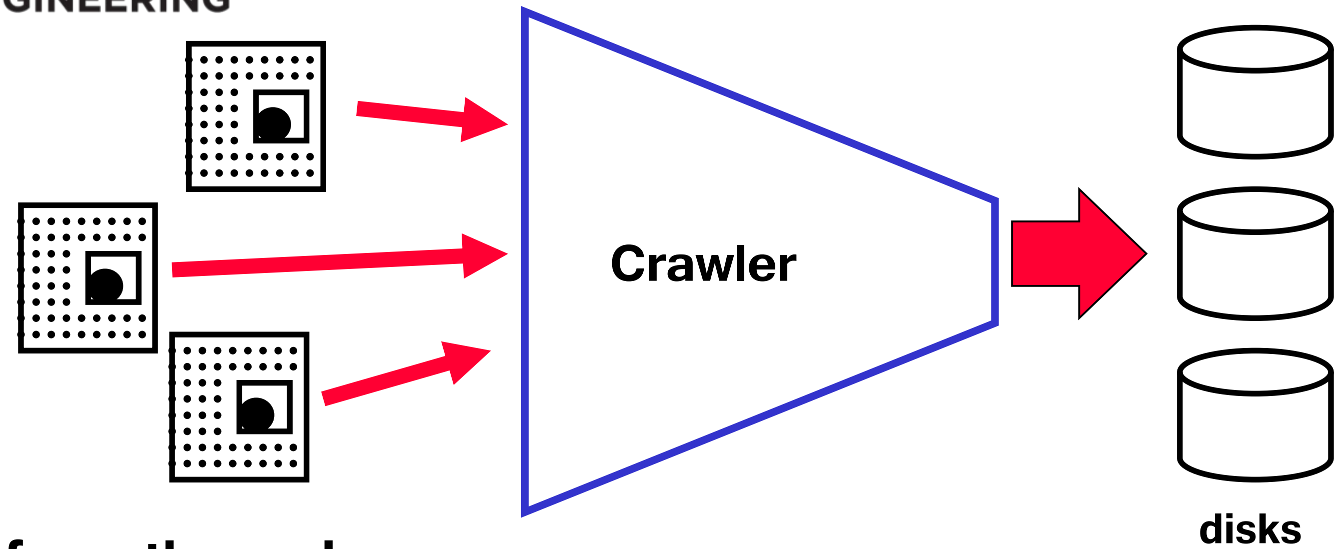


Four Major Components:

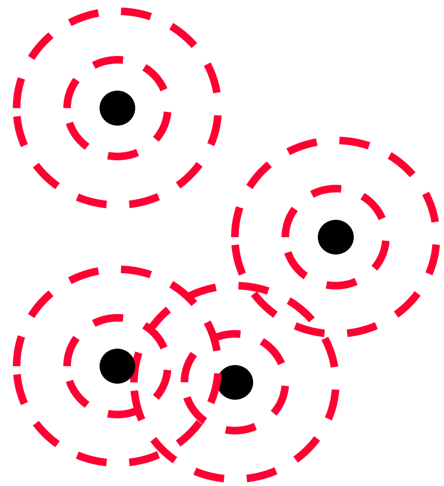
- data acquisition / web crawling
  - to collect pages from all over the web
  - messy process with lots of hand tuning
- data analysis / web mining
  - link analysis, spam detection, query log analysis
  - using mapreduce or similar specialized platforms
- index building
- query processing

most of the cycles spent in data analysis and query proc

Crawling:

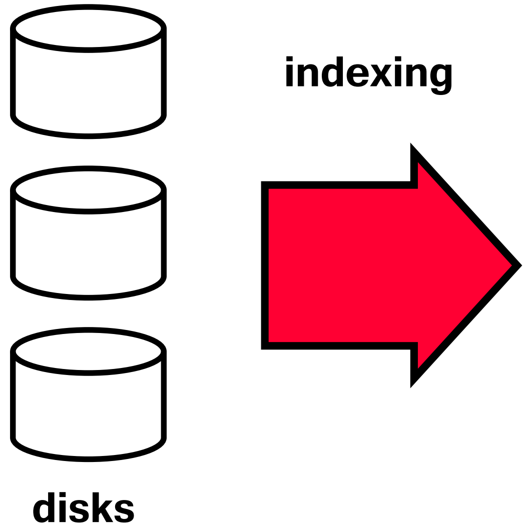


- fetches pages from the web
- starts at set of "seed pages"
- parses fetched pages for hyperlinks
- then follows those links (e.g., BFS)



- variations:
  - recrawling
  - focused crawling
  - crawl priorities to focus on important pages or to balance load

Indexing:



aardvark	3452, 11437, ...
.	
.	
.	
.	
.	
arm	4, 19, 29, 98, 143, ...
armada	145, 457, 789, ...
armadillo	678, 2134, 3970, ...
armani	90, 256, 372, 511, ...
.	
.	
.	
.	
.	
zebra	602, 1189, 3209, ...

"inverted index"

- parse & build lexicon & build index
- indexes are very large

→ I/O-efficient and parallel algorithms needed

Querying:

Boolean queries: ***(dog AND cat) OR mouse***

→ **compute unions/intersections of lists**

Ranked queries: ***zebra armadillo***

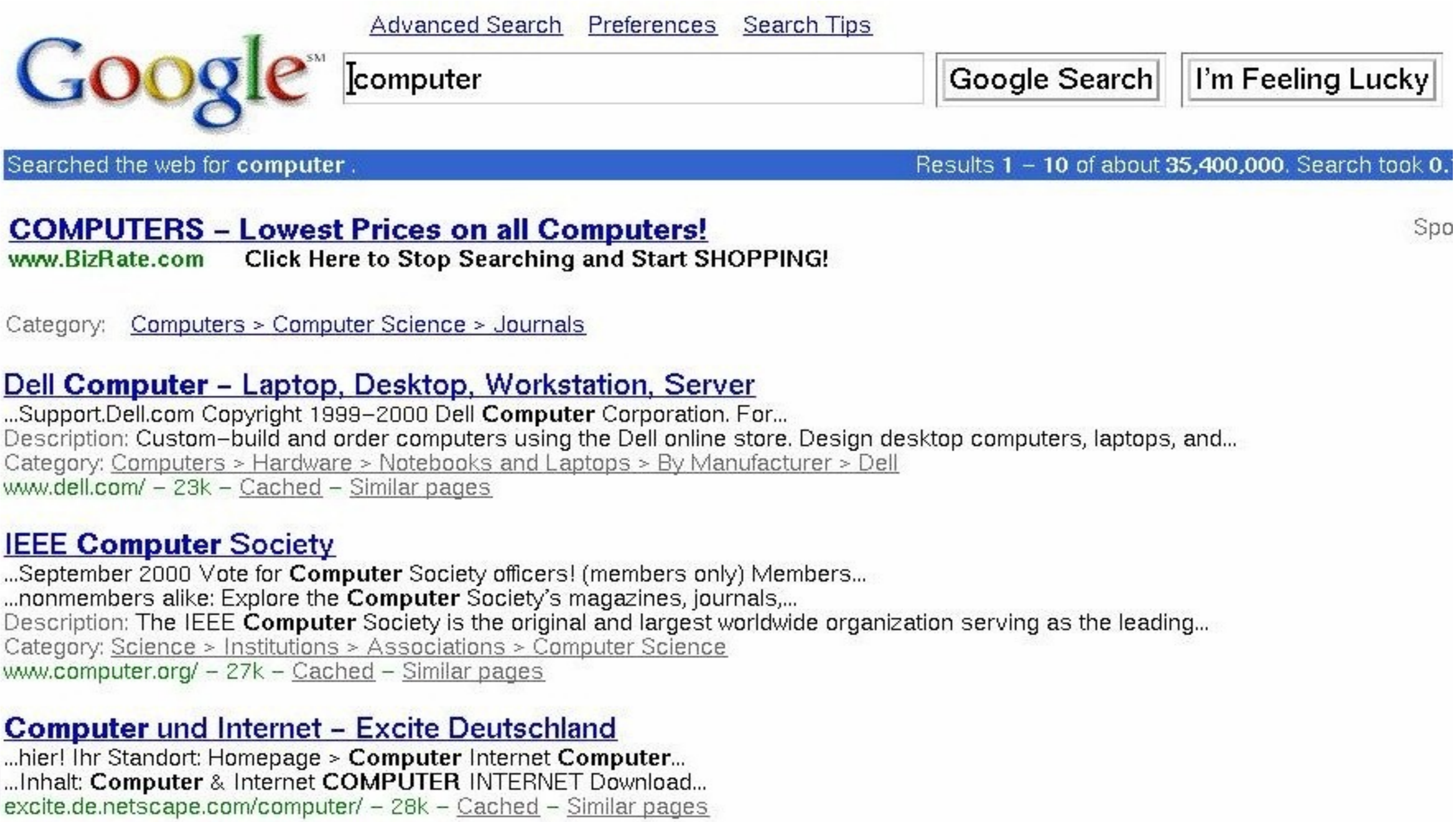
→ **give scores to all docs in union/intersect**

aardvark	3452, 11437, ...
.	
.	
.	
.	
.	
arm	4, 19, 29, 98, 143, ...
armada	145, 457, 789, ...
armadillo	678, 2134, 3970, ...
armani	90, 256, 372, 511, ...
.	
.	
.	
.	
.	
zebra	602, 1189, 3209, ...

look up

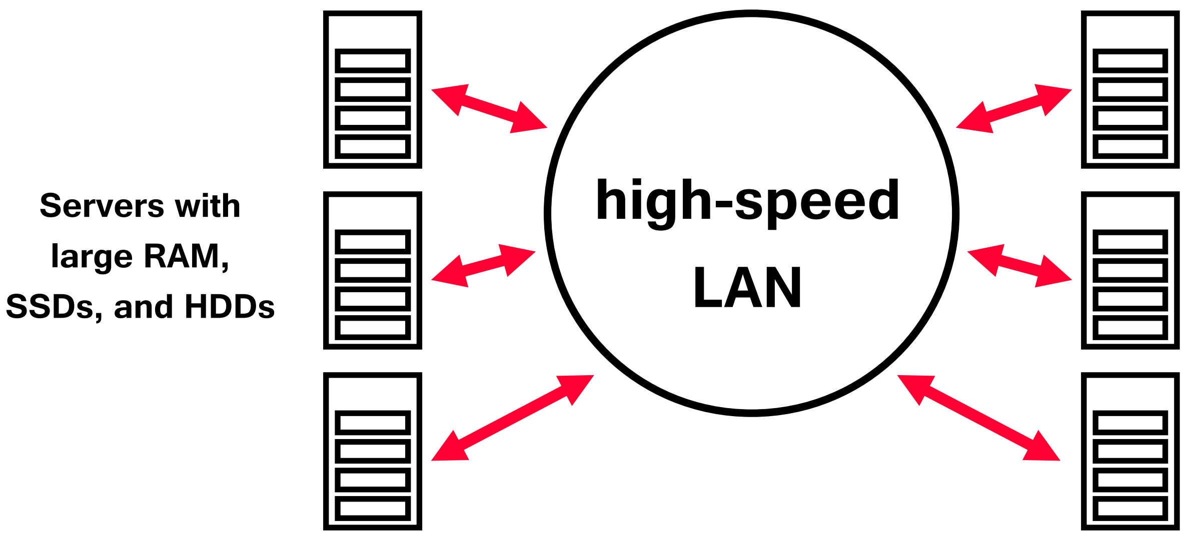
Ranking:

- return best pages first
- term, link, and click-based approaches
- machine-learned ranking
- also add meaningful snippets



Hardware setup:

- large data centers running customized Linux
- document and index data partitioned over many machines
- no ACID *(E. Brewer: “BASE = Basically Available, Soft-state, Eventual consistency”)*
- not a (relational) database



Hardware setup:

- large data centers running customized Linux
- document and index data partitioned over many machines
- no ACID *(E. Brewer: "BASE = Basically Available, Soft-state, Eventual consistency")*
- not a (relational) database

- search system challenges led to new generation of software tools:
  - distributed file systems such as Google FS, Hadoop FS
  - tuple stores and noSQL DBs (Cassandra, Hbase, Mongo, BigTable, Sherpa)
  - data analysis tools (Hadoop mapreduce, Pig, Spark, Giraph)
  - big impact on emerging data center architectures and energy management techniques

NYU | TANDON SCHOOL OF ENGINEERING

Major search engines are based on scalable clusters of low-cost servers connected by LANs

- many data centers with 10000s of servers each (Google)
- > trillions of web pages and many millions of web sites
- need to crawl, store, and process petabytes of data
- >> 50000 queries / second (major engines)
- "giant-scale" or "planetary-scale" web service

  (google search, facebook, gmail, twitter, wechat)
- but a lot of proprietary code and secret recipes

Google Reincarnates Dead Paper Mill as Data Center of Future

BY CADE METZ 01.26.12 6:30 AM

Follow @cademetz

Like 366 | Tweet 786 | +1 267 | Share 198

Google to expand data center in Council Bluffs

By Andrew J. Nelson
WORLD-HERALD STAFF WRITER

+1 | Like 4 | Share

ng. They just didn't expect it this

l Bluffs was completed, the
hat would roughly double its

tment in Iowa to about $1.1 billion
ouncil Bluffs data center.

Iceland Gets Major Data Center Project

By: Rich Miller
January 18th, 2010

Like 25 | Tweet | +1 | Share | Print

Apple Eyes First Overseas Data Cente in Hong Kong

BY CADE METZ 11.19.12 10:12 AM

Follow @cademetz

Nation-World

Google to open $150 million data center in South America

Simeon Tegel | GlobalPost.com | Nov 27, 2012

ALASKA NEWS & FEATURES

Bird Creek potter celebrates 33 years

LIMA, Peru — News that Google will open a $150 million data center in Chile has highlighted Latin America's IT paradox.

Executives at the internet's pre-eminent search engine expect to have the state-of-the-art facility, powered entirely by renewable energy and employing 20 people, up and running in the town of Quilicura, near the Chilean capital of Santiago, in 2013.

In a statement, Google said: "As internet usage in Latin America grows, people are looking for information and entertainment, new business opportunities and better ways to connect with friends and family near and far."

NYU-poly

POLYTECHNIC INSTITUTE OF NYU

Lecture 1: Introduction

I. The Web and Web Search:

- how the web works
- search tools

II. Introduction to Information Retrieval

- origins of IR
- basic setups and techniques

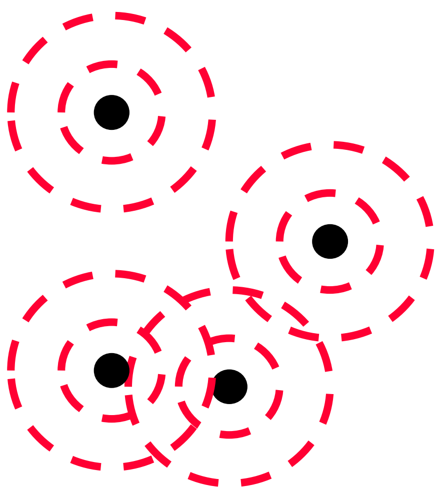
III. Main Components of a Search Engine

- basic search engine architecture
- web crawling
- indexing
- querying and ranking

NEW YORK UNIVERSITY

Web Crawling:

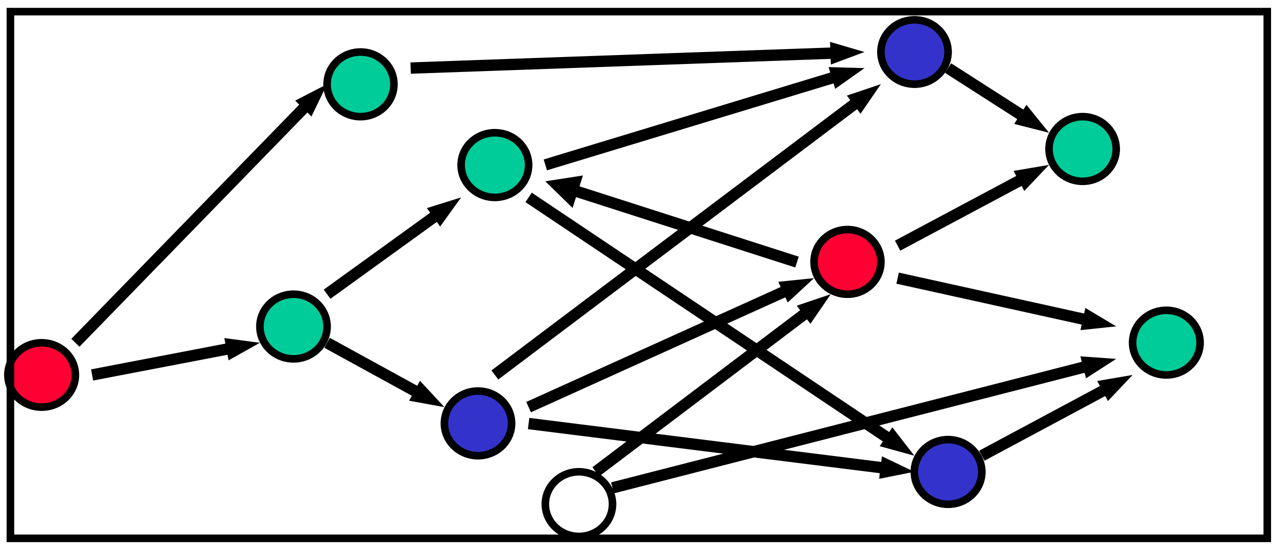
- **Basic idea:**
  - start at a set of known URLs
  - explore the web in "concentric circles" around these URLs

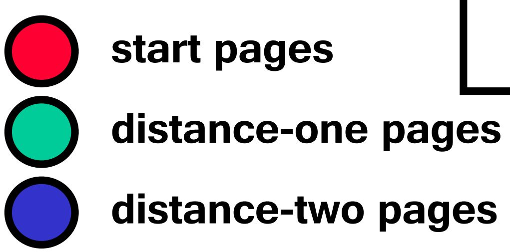


NYU | TANDON SCHOOL OF ENGINEERING

Web Crawling:

- **Basic idea:**
  - start at a set of known URLs
  - explore the web in "concentric circles" around these URLs





Simple Breadth-First Search Crawler:

```
insert set of initial URLs into a queue Q
while Q is not empty
    currentURL = dequeue(Q)
    download page from currentURL
    for any hyperlink found in the page
        if hyperlink is to a new page
            enqueue hyperlink URL into Q
```

this will eventually download all pages reachable from the start set

(also, need to remember pages that have already been downloaded)

Traversal strategies: (why BFS?)

- crawl will quickly spread all over the web
- load-balancing between servers
- in reality, more refined strategies *(but still somewhat BFSish)*
- many other strategies *(focused crawls, recrawls, site crawls)*

Tools/languages for implementation:

- Scripting languages *(Python, Perl)*
- Java *(performance tuning can be tricky)*
- C/C++ with sockets *(low-level)*
- available crawling tools *(usually not completely scalable)*

Details: *(lots of 'em)*

- **handling filetypes**

  (exclude some extensions, and use mime types)

- **URL extensions and CGI scripts**

  (to strip or not to strip? Ignore?)

- **frames, imagemaps, base tags**

- **black holes (robot traps, spam bots)**

  (limit maximum depth of a site)

- **different names for same site**

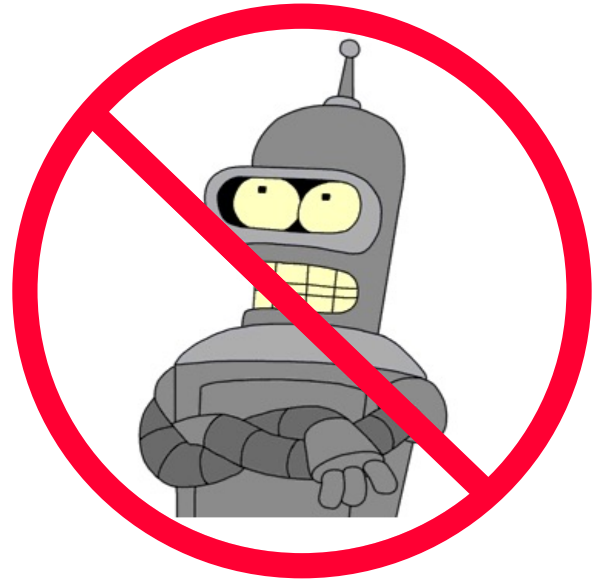
  (could check IP address, but no perfect solution)

- **duplicates, mirrors**

Performance considerations: later!

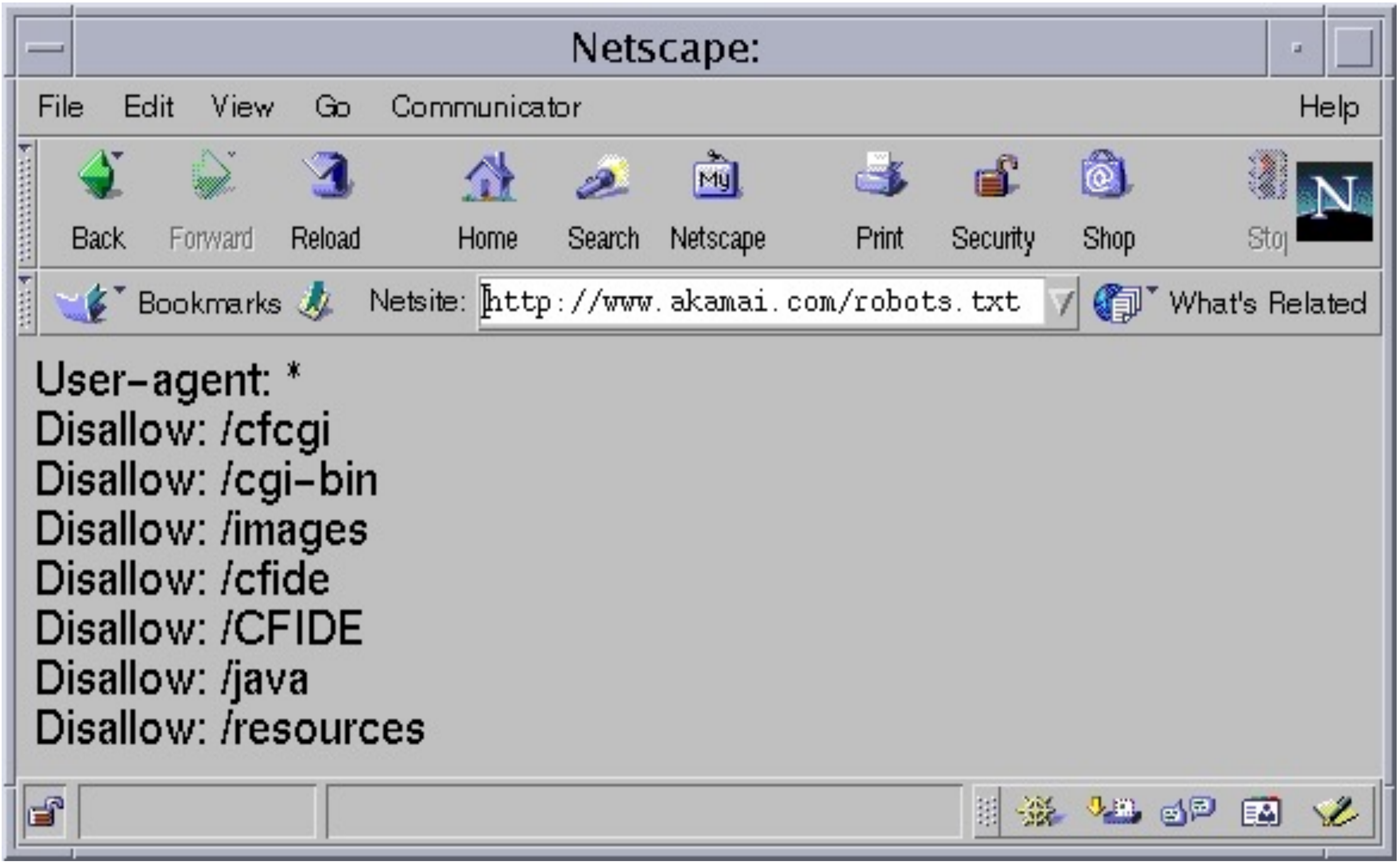
NYU | TANDON SCHOOL OF ENGINEERING

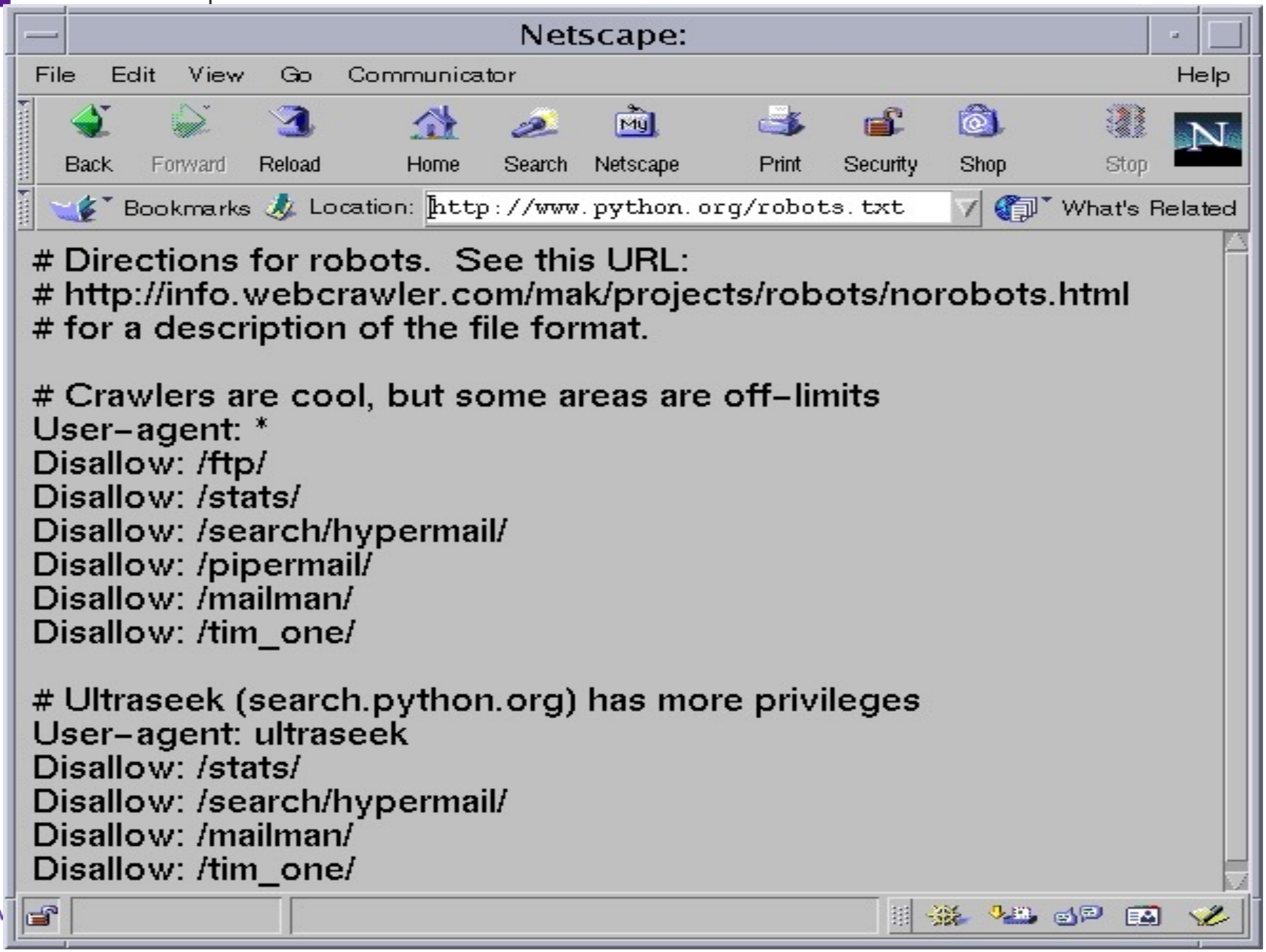
Robot Exclusion Protocol



- file robots.txt in root directory
- allows webmaster to "exclude" crawlers *(crawlers do not have to obey)*
- may exclude only certain robots or certain parts of the site
  - to "protect proprietary data" (e.g., eBay case)
  - to prevent crawlers from getting lost
  - to avoid load due to crawling
  - to avoid crashes (protect CGI bin)
- also allows to set crawl delay
- details at https://yoast.com/ultimate-guide-robots-txt/
- if at all possible, follow robot exclusion protocol!

Robot exclusion - example:






```
# Directions for robots.  See this URL:
# http://info.webcrawler.com/mak/projects/robots/norobots.html
# for a description of the file format.

# Crawlers are cool, but some areas are off-limits
User-agent: *
Disallow: /ftp/
Disallow: /stats/
Disallow: /search/hypermail/
Disallow: /pipermail/
Disallow: /mailman/
Disallow: /tim_one/

# Ultraseek (search.python.org) has more privileges
User-agent: ultraseek
Disallow: /stats/
Disallow: /search/hypermail/
Disallow: /mailman/
Disallow: /tim_one/
```

NYU | TANDON SCHOOL OF ENGINEERING

Robot META Tags

- allow page owners to restrict access to pages
- does not require access to root directory
- excludes all robots
- not supported by all crawlers, rarely used
- "noindex" and "nofollow"
- rarely used

Crawling Courtesy

- minimize load on crawled server
- no more than one outstanding request per site
- better: wait some seconds between accesses to same site

  (this number is not fixed)
- problems:
  - one server may have many sites *(use domain-based load-balancing)*
  - one site may have many pages *(3 years to crawl 3-million page site)*
  - intervals between requests should depend on site
- give contact info for very large crawls (email or URL)
- expect to be contacted ...

NYU | TANDON SCHOOL OF ENGINEERING

Crawling Challenges

- crawler may have to run for several weeks or months
- will interact with millions of web server
- some of them will be odd:
  - noncompliant server responses
  - unfamiliarity with robot exclusion protocol
  - robot traps
  - CGI and unintended consequences
  - network security tools
  - weird webmasters
- unclear legal situation for crawling (in 3 ways)

Lecture 1: Introduction

I. The Web and Web Search:

- how the web works
- search tools

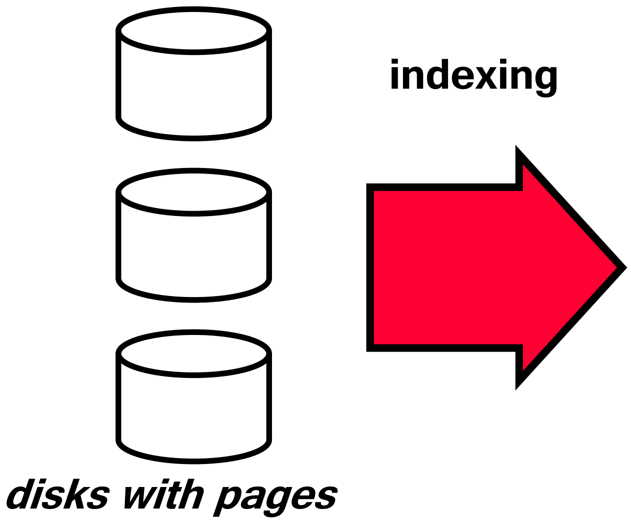
II. Introduction to Information Retrieval

- origins of IR
- basic setups and techniques

III. Main Components of a Search Engine

- basic search engine architecture
- web crawling
- indexing
- querying and ranking

Indexing:



aardvark	3452, 11437, ...
.	
.	
.	
.	
.	
arm	4, 19, 29, 98, 143, ...
armada	145, 457, 789, ...
armadillo	678, 2134, 3970, ...
armani	90, 256, 372, 511, ...
.	
.	
.	
.	
.	
zebra	602, 1189, 3209, ...

inverted index

- would like to build an index
  - in I/O-efficient manner if index cannot fit in main memory
  - in parallel on many machines and using many cores
- closely related to sorting
- also, can we compress an index *(ideally while building it)*

Indexing:

aardvark	3452, 11437, ...
.	
.	
.	
.	
arm	4, 19, 29, 98, 143, ...
armada	145, 457, 789, ...
armadillo	678, 2134, 3970, ...
armani	90, 256, 372, 511, ...
.	
.	
.	
.	
zebra	602, 1189, 3209, ...

inverted index

- *index entry* or *index posting*
  - usually contains document ID and frequency of term in document
  - or a precomputed term impact score and sometimes even position info
- *inverted list*: all index entries for one term
- document IDs can be assigned in various ways

Basic Indexing Concepts and Choices:

- we are interested in building a “full-text index”
- meaning: all words/terms that occur in the doc are indexed
- alternative: “keyword index” -- only index certain words
  - e.g., words in title or abstract, keywords provided by publisher
  - words that are identified as important via text mining or NLP
  - or other features/terms that we want to associate the document with

NYU | TANDON SCHOOL OF ENGINEERING

Basic Indexing Concepts and Choices:

- **we are interested in building a "full-text index"**
- **meaning: all words/terms that occur in the doc are indexed**
- **alternative: "keyword index" -- only index certain words**
  - e.g., words in title or abstract, keywords provided by publisher
  - words that are identified as important via text mining or NLP
  - or other features/terms that we want to associate the document with
- **what is a word?**

NYU | TANDON SCHOOL OF ENGINEERING

Basic Indexing Concepts and Choices:

- we are interested in building a “full-text index”
- meaning: all words/terms that occur in the doc are indexed
- alternative: “keyword index” -- only index certain words
  - e.g., words in title or abstract, keywords provided by publisher
  - words that are identified as important via text mining or NLP
  - or other features/terms that we want to associate the document with
- what is a word?
  - anything between two spaces
  - or between two separating characters such as , ; . ( ) - [ ] ? and so on
  - includes numbers, misspelled words, garbage (maybe limit length)
  - how do we separate words e.g., in Chinese?
  - is “New York City” one word or three?
- lexicon: set of all words encountered in collection

Basic Indexing Concepts and Choices:

- for each word occurrence:

  store index of document where it occurs, and frequency of word in document

- also store position within document? *(often yes)*
  - increases space for index significantly!
  - allows efficient search for phrases
  - relative positions of words may be important for ranking
  - otherwise, need to scan document to determine positions
- also store additional context of words? *(in title, bolded, in URL, in anchortext)*
- *stop words*: common words such as "is", "a", "the"
- ignore stop words? *(maybe better not)*
  - saves space in index
  - cannot search for "to be or not to be" or "the who"
  - queries with stop words expensive - more important than space issues
- stemming: *"runs = run = running"* *(depends on language)*
- parsing / tokenization issues

Indexing: *(simplified approach)*

(1) scan through all documents

(2) for every work encountered

generate entry (word, doc#, pos)

(3) sort entries by (word, doc#, pos)

(4) now transform into final form

doc1: "Bob reads a book"
doc2: "Alice likes Bob"
doc3: "book"

↓

bob, 1, 1 | reads, 1, 2 | a, 1, 3
book,1, 4 | alice, 2, 1 | likes, 2, 2
bob, 2, 3 | book, 3, 1

↓

a, 1, 3 | alice, 2, 1 | bob, 1, 1
bob, 2, 3 | book, 1, 4 | book, 3, 1
likes, 2, 2 | reads, 1, 2

↓

a:	(1,3)
Alice:	(2, 1)
Bob:	(1, 1), (2, 3)
book:	(1, 4), (3, 1)
likes:	(2, 2)
reads:	(1, 2)

NYU TANDON SCHOOL OF ENGINEERING

A Naive Approach

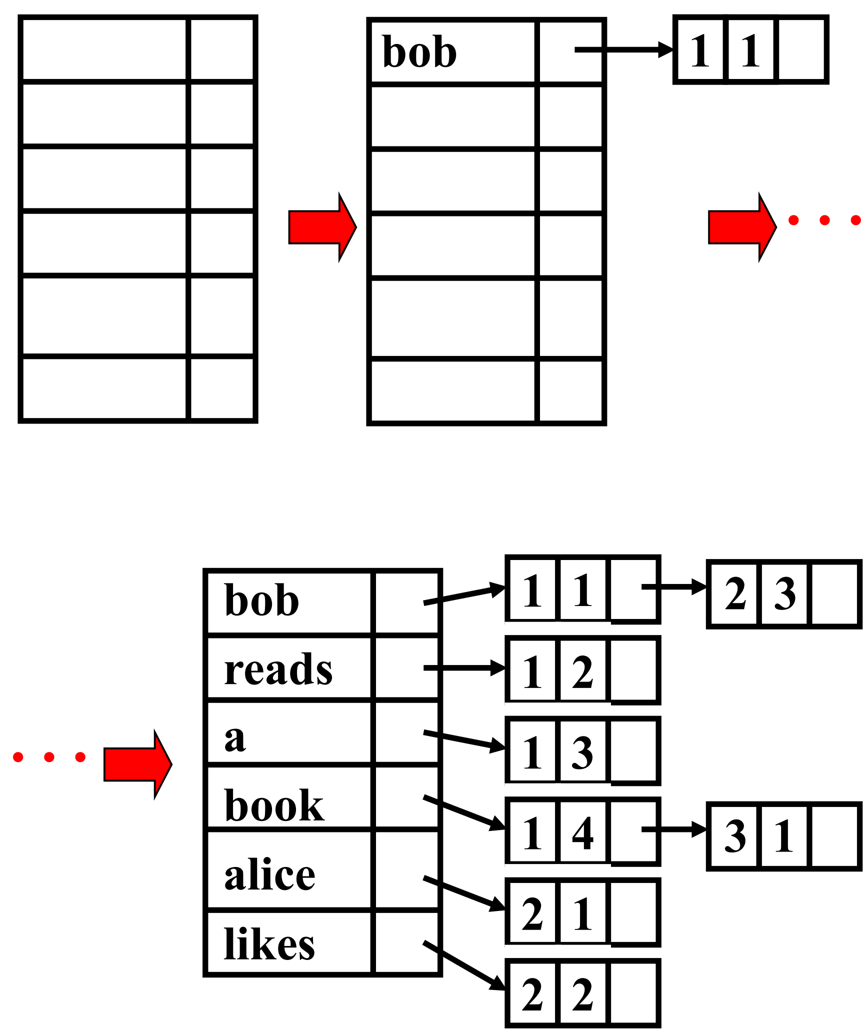
a) create an empty dynamic data structure (e.g., hash table) in main memory;

b) scan through all documents, for every word encountered:

- **i.** create an entry in dictionary if the word does not exist;
- **ii.** Insert (doc id, pos) into the inverted list corresponding to the word;

c) traverse the dictionary and dump inverted index on disk

doc1: "Bob reads a book"
doc2: "Alice likes Bob"
doc3: "book"



A Naive Approach

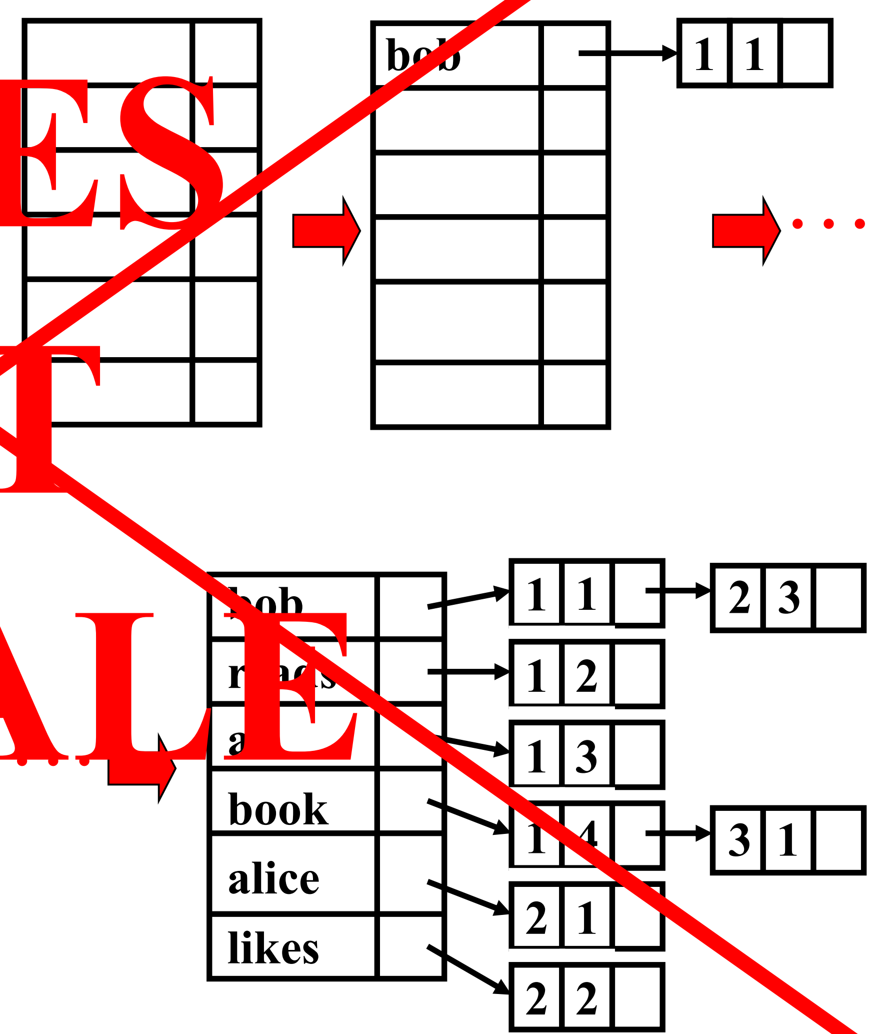
a) create an empty dynamic data structure (e.g. hash table) in main memory

b) scan through all documents, for every word encountered:

- **i.** create an entry in dictionary if the word does not exist;
- **ii.** Insert (doc id, pos) into the inverted list corresponding to the word;

c) traverse the dictionary and dump inverted index on disk

doc1: "Bob reads a book"
doc2: "Alice likes Bob"
doc3: "book"



DOES NOT SCALE

- **If data too large for memory then we need to implement indexing carefully to exploit properties of hard disk or SSD**
  - **would like to index thousands of pages per second**
  - **need to store pages on disk in consecutive order (we cannot afford separate seek time for each page)**
  - **need to implement highly efficient sort of large files of postings on disk (need to sort many MB of data per second)**
  - **resulting inverted index needs to be laid out sequentially to allow efficient query processing**
  - **Ideally integrate compression into indexing process as well as the final index structure format**
- **More details on how to do this in next lectures**

Inverted index compression

- suppose we only store document IDs (no position or context)
- should we use 32-bit integer for each document ID?

:	
arm	4, 19, 29, 98, 143, ...
armada	145, 457, 789, ...
armadillo	678, 2134, 3970, ...
armani	90, 256, 372, 511, ...
:	

:	
arm	4, 15, 10, 69, 45, ...
armada	145, 312, 332, ...
armadillo	678, 1456, 1836, ...
armani	90, 166, 116, 139, ...
:	

- encode sorted runs by their gaps
  - significant compression for frequent words!
- we can also compress frequencies
- many highly optimized schemes studied
- can decompress billions of postings per second

Lecture 1: Introduction

I. The Web and Web Search:

- how the web works
- web search tools

II. Introduction to Information Retrieval

- origins of IR
- basic setups and techniques

III. Main Components of a Search Engine

- basic search engine architecture
- web crawling
- indexing
- querying and ranking

Querying:

aardvark	3452, 11437, …..
.	
.	
.	
.	
arm	4, 19, 29, 98, 143, ...
armada	145, 457, 789, ...
armadillo	678, 2134, 3970, ...
armani	90, 256, 372, 511, ...
.	
.	
.	
.	
.	
zebra	602, 1189, 3209, ...

Boolean queries: ***(dog AND cat) OR mouse***

→ **compute unions/intersections of lists**

Ranked queries: ***zebra armadillo***

→ **give scores to all docs in union/intersect**

look up

NYU | TANDON SCHOOL OF ENGINEERING

- most web queries involve one or two common words

  → Boolean querying returns millions of hits

- would like to rank results by ...
  - importance?
  - relevance?
  - accuracy?

- in general, arbitrary *score function*:

  "return pages with highest score relative to query"

- use inverted index as access path for pages
  - start with (possibly expanded) Boolean query (e.g., OR)
  - only rank Boolean results
  - in fact, try to avoid computing complete Boolean results (pruning methods, covered much later in semester)

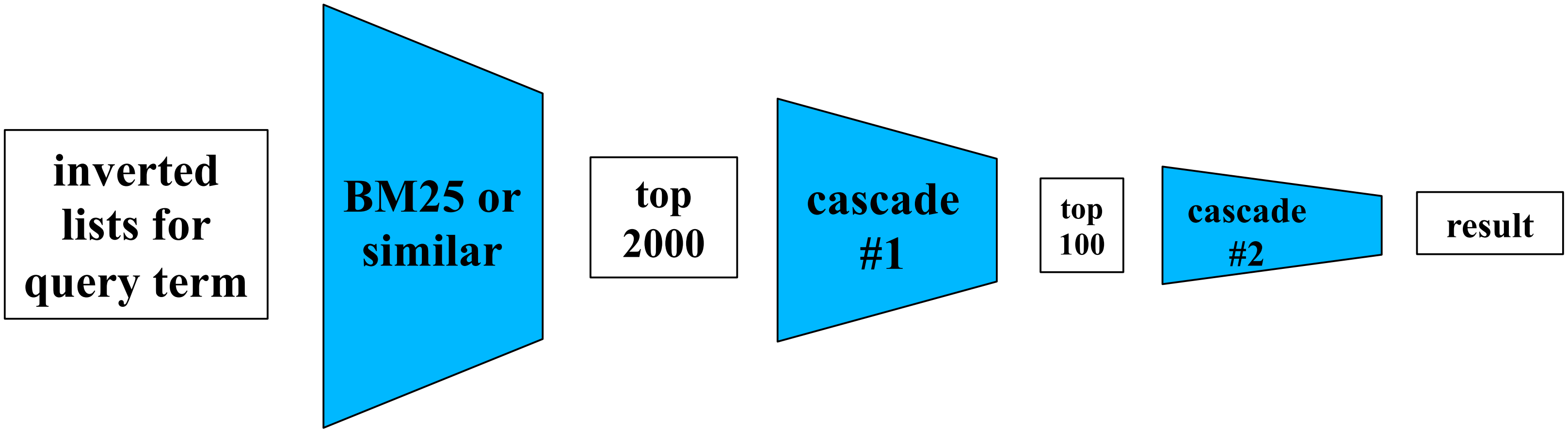
- **scoring function: assigns score to each document with respect to a given query**
- **top-k queries: return k documents with highest scores**
- **example cosine measure for query with terms** t_0 **to** t_{m-1}

$$
\begin{aligned}
F(d, t_0, \ldots, t_{m-1}) &= \sum_{i=0}^{m-1} \frac{w(q,t_i) \cdot w(d,t_i)}{\sqrt{|d|}}, \\
w(q,t) &= \ln(1 + N/f_t), \text{and} \\
w(d,t) &= 1 + \ln f_{d,t},
\end{aligned}
$$

- **can be implemented by computing score for all documents that contain any of the query words** ***(union of inverted lists)***
- **in case of search engines: often intersection instead of union**
- **in large collections, lists are many MB** ***for average queries***

Cascading Query Execution

- Everybody now uses highly complex ranking functions
- But these are expensive to compute
- Cascading approach (e.g., Wang/Lin/Metzler, SIGIR 2011)



- Each cascade trained on output of previous
- Simple ranking function = cascade #0 (candidate generation)

NYU | TANDON SCHOOL OF ENGINEERING

Approximate Semester Overview

- Chapter 1: Introduction
- Chapter 2: Indexing
- Chapter 3: Query Processing
- Chapter 4: Data and Index Compression
- Chapter 5: Link Analysis & Computing with Web Graphs
- Chapter 6: Search Logs, Sessions, and Clicks
- Chapter 7: Advanced Crawling and Search Architecture
- Chapter 8: Search Evaluation and Quality
- Chapter 9: Query Expansion and Relevance Feedback
- Chapter 10: Information Retrieval Models
- Chapter 11: Learning to Rank and Neural IR
- Chapter 12: Computational Advertising and Spam