Intro to Search Engines

A Survey on Web Search

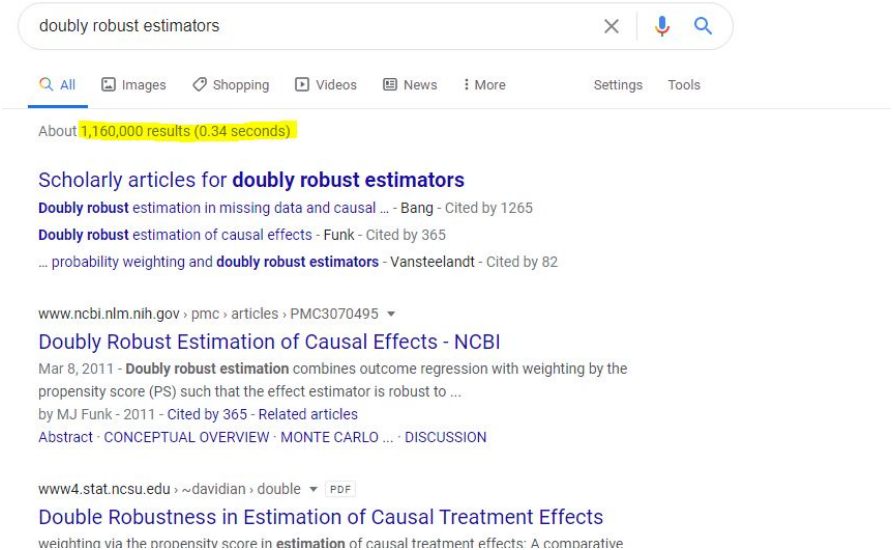
David Koleczek

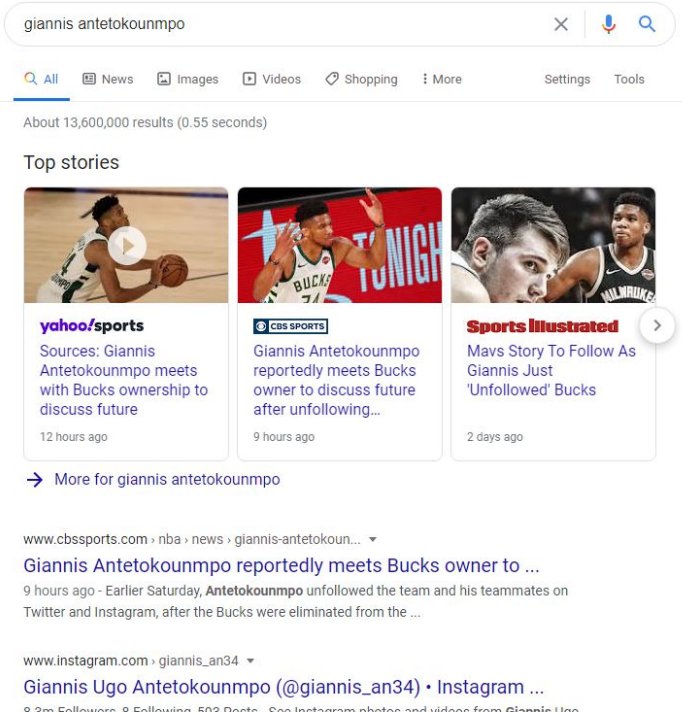
Introduction

- Focus here on web search and a survey of techniques
- Many concepts broadly applicable beyond web search
  - Large scale data processing, learning to rank, data structures

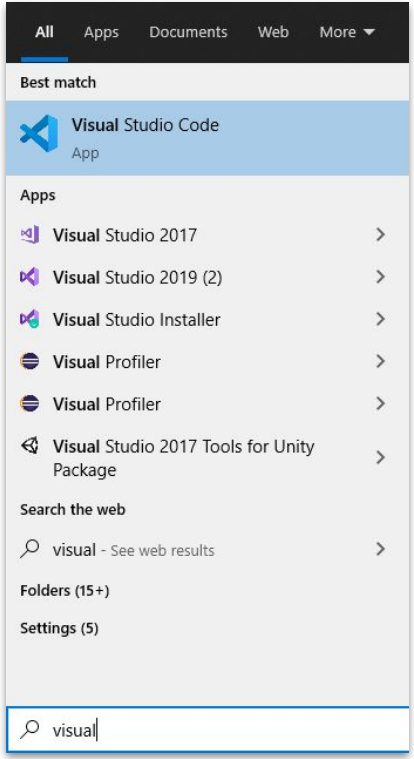
- Based loosely on Search Engines and Information Retrieval: Applications for Twitter
  - Which was inspired by UMass CS446 - Search Engines
- Search Engines Information Retrieval in Practice by Croft, Metzler, and Strohman

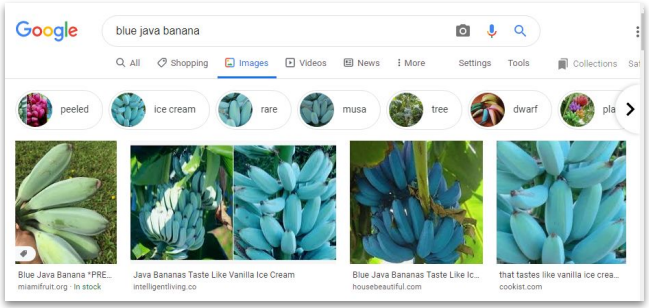
Motivation

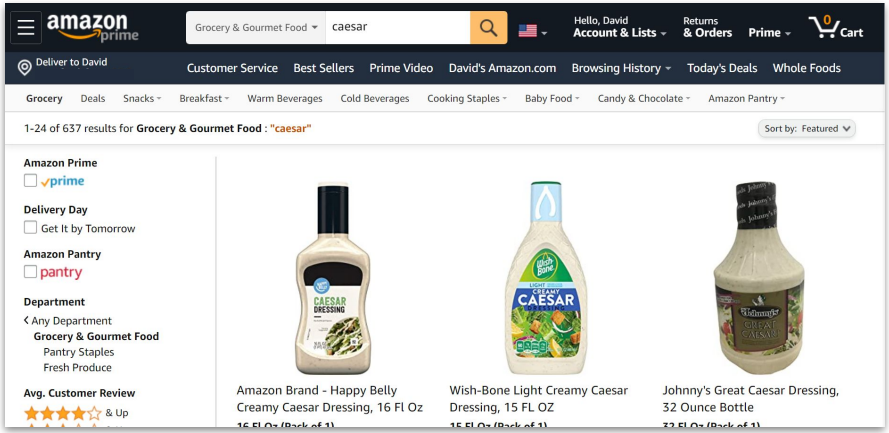

doubly robust estimators
All
Images
Shopping
Videos
News
More
Settings
Tools
About 1,160,000 results (0.34 seconds)
Scholarly articles for doubly robust estimators
Doubly robust estimation in missing data and causal ... - Bang - Cited by 1265
Doubly robust estimation of causal effects - Funk - Cited by 365
... probability weighting and doubly robust estimators - Vansteelandt - Cited by 82
www.ncbi.nlm.nih.gov › pmc › articles › PMC3070495
Doubly Robust Estimation of Causal Effects - NCBI
Mar 8, 2011 - Doubly robust estimation combines outcome regression with weighting by the propensity score (PS) such that the effect estimator is robust to ...
by MJ Funk - 2011 - Cited by 365 - Related articles
Abstract · CONCEPTUAL OVERVIEW · MONTE CARLO ... · DISCUSSION
www4.stat.ncsu.edu › ~davidian › double
PDF
Double Robustness in Estimation of Causal Treatment Effects
weighting via the propensity score in estimation of causal treatment effects: A comparative


giannis antetokounmpo
All
News
Images
Videos
Shopping
More
Settings
Tools
About 13,600,000 results (0.55 seconds)
Top stories
yahoo!sports
Sources: Giannis Antetokounmpo meets with Bucks ownership to discuss future
12 hours ago
CBS SPORTS
Giannis Antetokounmpo reportedly meets Bucks owner to discuss future after unfollowing...
9 hours ago
Sports Illustrated
Mavs Story To Follow As Giannis Just 'Unfollowed' Bucks
2 days ago
More for giannis antetokounmpo
www.cbssports.com › nba › news › giannis-antetokoun...
Giannis Antetokounmpo reportedly meets Bucks owner to ...
9 hours ago - Earlier Saturday, Antetokounmpo unfollowed the team and his teammates on Twitter and Instagram, after the Bucks were eliminated from the ...
www.instagram.com › giannis_an34
Giannis Ugo Antetokounmpo (@giannis_an34) • Instagram ...

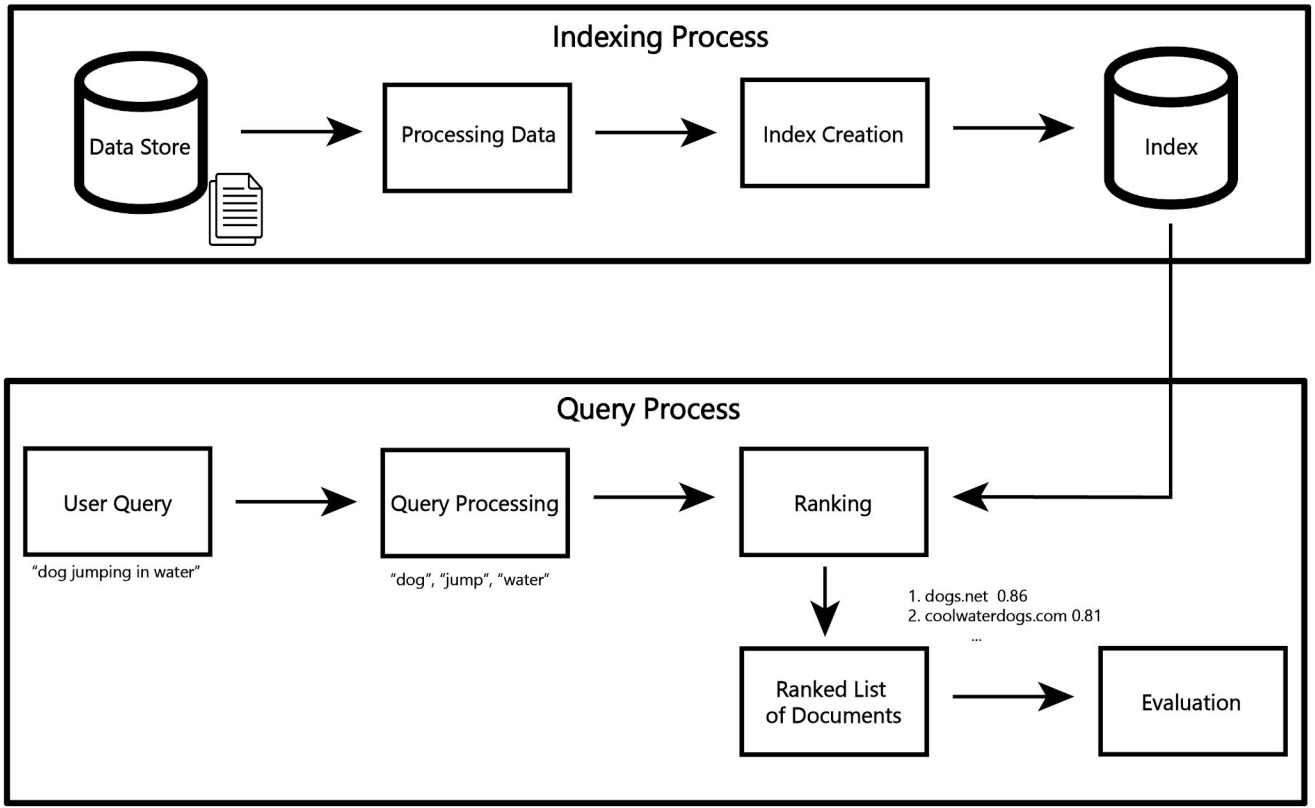
Information Retrieval: Not Just Web Search



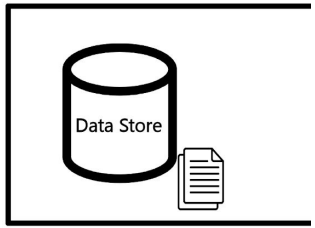


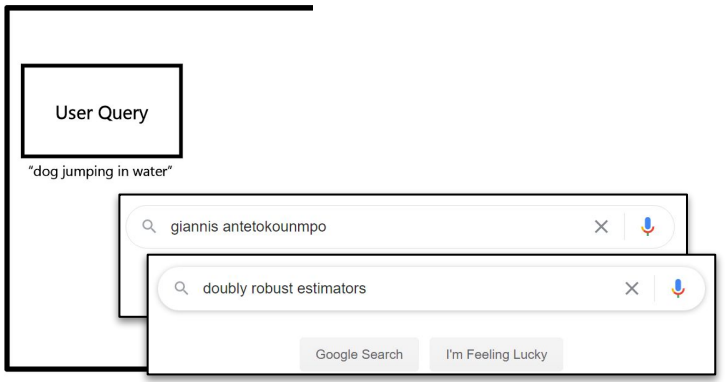


Architecture of a Search Engine



Search is a Data Problem



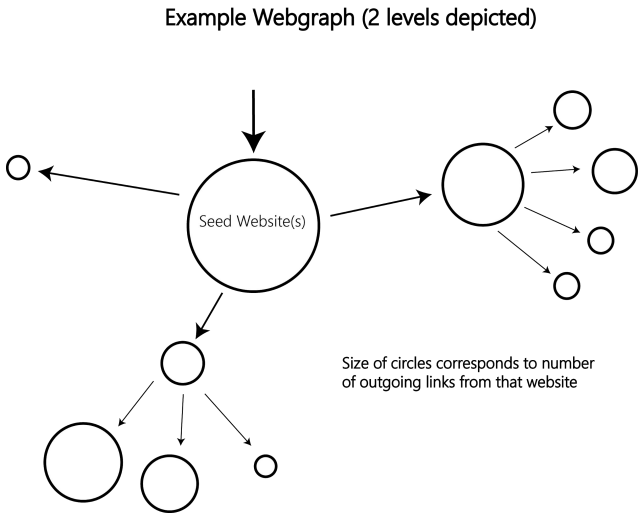



```
<DOC>
<DOCNO>WTX001-B01-10</DOCNO>
<DOCHDR>
http://www.example.com/test.html 204.244.59.33 19970101013145 text/html 440
HTTP/1.0 200 OK
Date: Wed, 01 Jan 1997 01:21:13 GMT
Server: Apache/1.0.3
Content-type: text/html
Content-length: 270
Last-modified: Mon, 25 Nov 1996 05:31:24 GMT
</DOCHDR>
<HTML>
<TITLE>Tropical Fish Store</TITLE>
Coming soon!
</HTML>
</DOC>
<DOC>
<DOCNO>WTX001-B01-109</DOCNO>
<DOCHDR>
http://www.example.com/fish.html 204.244.59.33 19970101013149 text/html 440
HTTP/1.0 200 OK
Date: Wed, 01 Jan 1997 01:21:19 GMT
Server: Apache/1.0.3
Content-type: text/html
Content-length: 270
Last-modified: Mon, 25 Nov 1996 05:31:24 GMT
</DOCHDR>
<HTML>
<TITLE>Fish Information</TITLE>
This page will soon contain interesting
information about tropical fish.
</HTML>
</DOC>
```

Example of “raw” web pages

Interlude: How to find and store 30 trillion websites?

Web Crawling



Mostly a solved problem, see Apache Nutch or Scrapy's web spiders

Interlude: How to find and store 30 trillion websites?

Crawling happens ALL the time

```
207.46.13.215 - - [28/Mar/2020:12:31:14 +0000] "GET / HTTP/1.1" 200 1120 "-"
        "Mozilla/5.0 (compatible;bingbot/2.0; +http://www.bing.com/bingbot.htm)"

66.249.68.30 - - [20/May/2020:14:39:11 +0000] "GET /robots.txt HTTP/1.1" 200 67 "-"
        "Mozilla/5.0 (compatible;Googlebot/2.1; +http://www.google.com/bot.html)"
66.249.68.30 - - [20/May/2020:14:55:22 +0000] "GET / HTTP/1.1" 200 1120 "-"
        "Mozilla/5.0 (Linux; Android 6.0.1; Nexus 5X Build/MMB29P) AppleWebKit/537.36 (KHTML, like Gecko)
        Chrome/80.0.3987.92 Mobile Safari/537.36 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)"

3.218.77.26 - - [31/Aug/2020:20:32:40 +0000] "GET /robots.txt HTTP/1.1" 200 67 "-"
        "Mozilla/5.0 (compatible; ia_archiver/1.0; +http://www.alexa.com/help/webmasters;crawler@alexa.com)"
3.218.77.26 - - [31/Aug/2020:20:32:40 +0000] "GET / HTTP/1.1" 200 1116 "-"
        "Mozilla/5.0 (compatible; ia_archiver/1.0; +http://www.alexa.com/help/webmasters; crawler@alexa.com)"

17.58.97.227 - - [31/Aug/2020:22:10:43 +0000] "GET / HTTP/1.1" 200 1116 "-"
        "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_5) AppleWebKit/605.1.15 (KHTML, like Gecko)
        Version/13.1.1 Safari/605.1.15 (Applebot/0.1; +http://www.apple.com/go/applebot)"
```

Sample weblog from a small web server

Interlude: How to find and store 30 trillion websites?

Google File System (later Google Colossus)

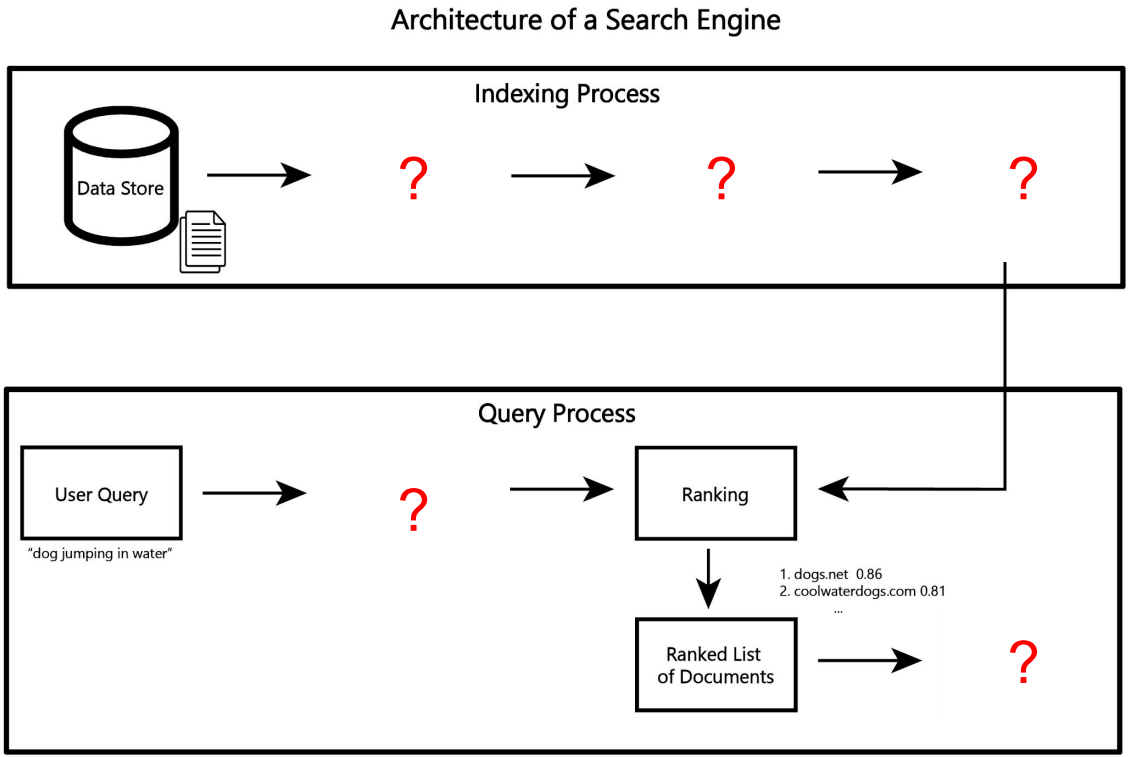
- Petabytes of web pages are stored across thousands of “cheap” servers
- Allow for constant scaling up to keep up with the growth of the web

Google Bigtable

- Built on top of the massive data stores
- Semi-structured storage system
  - Data is indexed in a bigtable using row and column names that can be arbitrary strings.
    - (row:string, column:string, time:int64) → string
    - row: URL of the website, column: “content” → HTML content of website
  - Data stored as uninterpreted strings, up to user to serialize their original data
- Adaptively spreads data across thousands of nodes

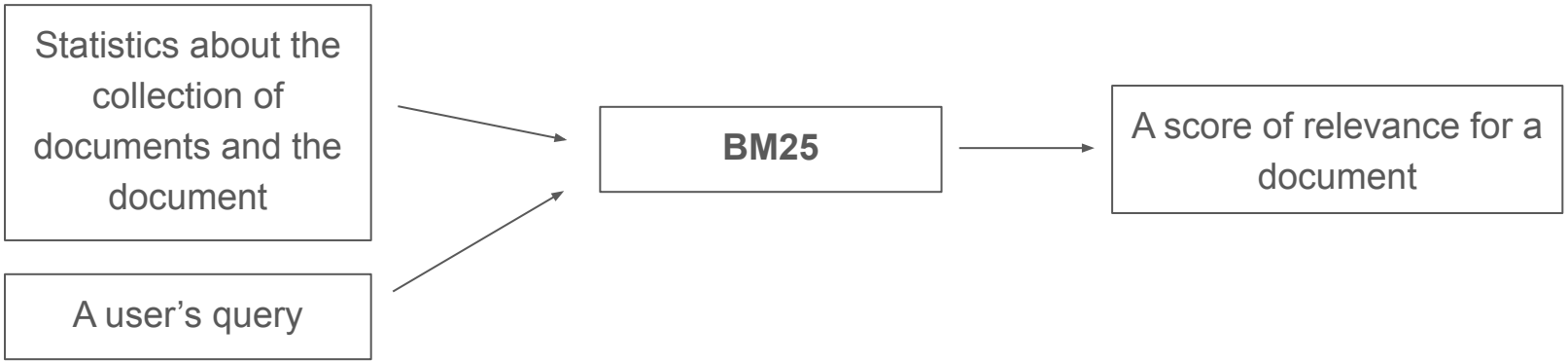
Ranking

- How do we rank websites given a collection of websites and a user query?
- Can be viewed as the modeling part of a search engine



Retrieval Models - BM25

- Estimates the relevance of a document with respect to a given search query



Retrieval Models - BM25

$$\sum_{i \in Q} \left(log\left(\frac{N - n_i + 0.5}{n_i + 0.5} \right) \left(\frac{(k_1 + 1) f_i}{K + f_i} \right) \left(\frac{(k_2 + 1) q f_i}{k_2 + q f_i} \right) \right)$$

- A summation over every term in the user's query
- Operates document at a time (run this formula for each document)

i := i^{th} term in tokenized query Q
N := number of docs in the collection
n_i := number of docs containing term i

k_1 := constant, hyperparameter
f_i := frequency of term i in the document

K := $k_1((1-b) + b(dl / avgdl))$
b := constant, hyperparameter
dl := length of the document
$avgdl$:= average length of a document in the collection

k_2 := constant, hyperparameter
qf_i := frequency of term i in the query

BM25: Inverse Document Frequency (*idf*)

$$log(\frac{N-n_i+0.5}{n_i+0.5})$$

i := i^{th} term in tokenized query Q
N := number of docs in the collection
n_i := number of docs containing term i

- First term in the BM25 summation is known as the *idf* component.
- Penalizes words in the query that occur in many documents
- If the number of documents containing a term, n_i, is 1
  - Will result in a very high value
  - However, if n_i is close to N, then we will have a very low value.
  - log “dampens” the effect
  - 0.5 prevents division by 0

BM25: Term Frequency (*tf*)

$$\left(\frac{(k_1+1)f_i}{K+f_i}\right)$$

k_1 := constant, hyperparameter, [1.2, 2]
f_i := frequency of term *i* in the document

$K := k_1((1-b) + b(dl / avgdl))$
b := constant, hyperparameter, 0.75
dl := length of the document
avgdl := average length of a document in the collection

- If we disregard the constants that are fixed for every document (k_1, *b*, *avgdl*) , we are left with: f_i / *dl* + f_i
  - Ratio of how frequent a query term is compared to the length of the document
  - Higher score the more frequent the term is in the document
  - Lower score the longer the document
- k_1 is a scaling factor for the entire component
  - The higher it is set, the more impact this term will have
- *b* regulates the length normalization, *dl* / *avgdl*
  - *b* = 0 means it is ignored, *b* = 1 is full normalization
- *avgdl* serves to increase the score if the document is shorter than average, and vice versa

BM25: Query Term Frequency (*qtf*)

$$\left(\frac{(k_2+1)qf_i}{k_2+qf_i}\right)$$

k_2 := constant, hyperparameter, [0, 1000]
qf_i := frequency of term *i* in the query

- Factors in how frequently terms appear in the **query**
- If $k_2 > 1$, as qf_i increases, its contribution to the score will also increase.

Retrieval Models - BM25

$$\sum_{i \in Q} \left(log\left(\frac{N - n_i + 0.5}{n_i + 0.5} \right) \left(\frac{(k_1+1) f_i}{K + f_i} \right) \left(\frac{(k_2+1) q f_i}{k_2 + q f_i} \right) \right)$$

idf *tf* *qtf*

- *idf* term penalizes words in the query that occur in **many** documents
- *tf* gives high scores to terms that occur frequently within a **single** document
- *qtf* gives higher scores to frequent query terms

Example Query: “Spotify Wikipedia”

i := i^{th} term in tokenized query Q
N := number of docs in the collection
n_i := number of docs containing term i

k_1 := constant, hyperparameter
f_i := frequency of term i in the document

K := $k_1((1-b) + b(dl / avgdl))$
b := constant, hyperparameter
dl := length of the document
$avgdl$:= average length of a document in the collection

k_2 := constant, hyperparameter
qf_i := frequency of term i in the query

Ranking Features

Many domain specific features can be derived to come up with a score of how relevant a document is to a query.

BM25 - mostly term frequency of documents, length of documents, “bag of words” statistics

PageRank - Measures importance of a webpage based on the number of links that point to that website

Recency - newer websites should likely be higher

Ranking Features

Core question: How to combine these different features to get one score per document?

General Process:

- Scale each feature, ex. each feature falls in the range [0,1]
- Take a weighted average

$$scaled\_sigmoid(x) = \frac{1}{1+((\frac{e}{c})+1)^{-x}}, \text{ where } c \text{ is some constant}$$

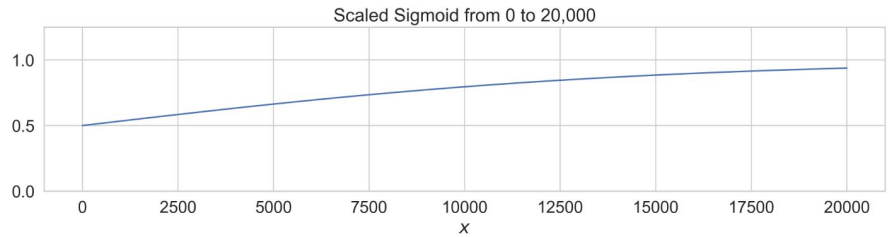


Figure 5: "Scaled sigmoid" with $c = 20,000$ plotted for values 0 to 20,000.

Interlude: Learning to Rank

Problem Formulation (Microsoft Learning to Rank Datasets)

- Each row is a query, url/document pair
- First column is the relevance label (0-4)
- Second column is the query id
- Remaining columns are feature:value

```
0 qid:1 1:3 2:0 3:2 4:2 ... 135:0 136:0

2 qid:1 1:3 2:3 3:0 4:0 ... 135:0 136:0
```

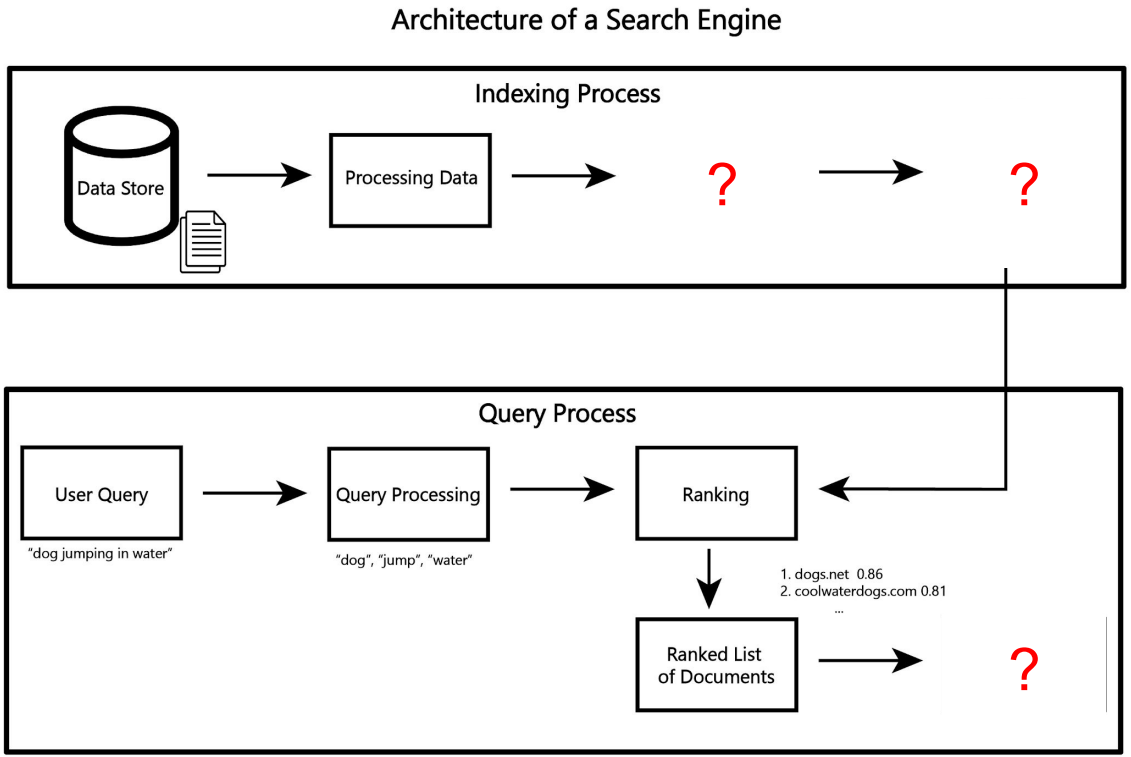
Two rows from MSLR-WEB10K dataset

Use machine learning to create a ranking

- **Learning to Rank**: Class of algorithms that optimize for metrics like mean average precision (MAP) or discounted cumulative gain (DCG)
  - As opposed to regression or classification metrics like MAE or cross-entropy

Many LTR implementations exist, including in LightGBM and XGBoost

Text Processing



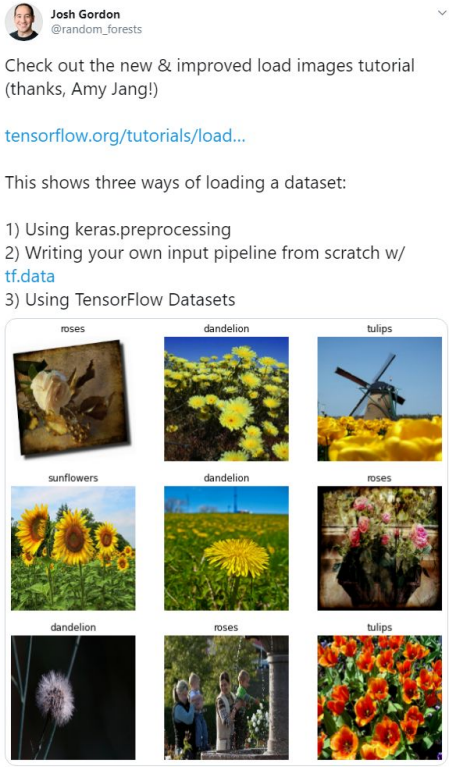
Text Processing

A Natural Language Processing (NLP) problem!

- For both documents **and** query understanding

Need to go from raw content of the document to features that work the best in our ranking function.

Text Processing - Example



['josh', 'gordon', 'check', 'new', 'amp', 'improve', 'load', 'image', 'tutorial', 'thank', 'amy', 'jang', 'show', 'way', 'load', 'dataset', 'keras.preprocesse', 'write', 'input', 'pipeline', 'scratch', 'w/', 'tensorflow', 'dataset', 'load', 'image', 'tensorflow', 'core', 'august', ‘aug’, '16', '2020']

- Append username
- Stop words - remove most common words like “the”
  - “to be or not to be”?
- Remove punctuation and symbols
- Lemmatization - converting tokens to their lemmas based on part of speech and potentially the context of the word
- Remove URLs and replace with the “resolved” website’s title and description (depends on if the website provides this easily)
- Append the date

Text Processing

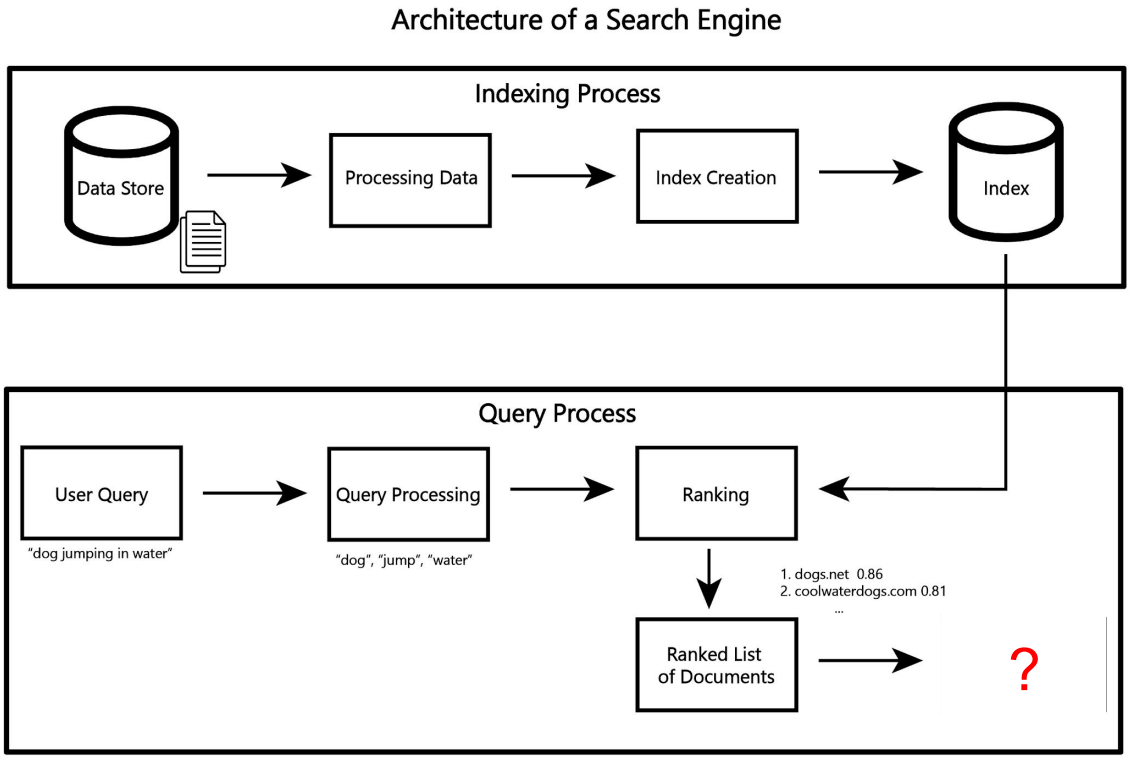
Handling the user query is as important as the data itself

- What did a user really mean?

Some basic techniques

- Spelling correction
- Replacing terms with synonyms
- Expansion by adding close matching words

Indexing


Architecture of a Search Engine
Indexing Process
Data Store
Processing Data
Index Creation
Index
Query Process
User Query
"dog jumping in water"
Query Processing
"dog", "jump", "water"
Ranking
1. dogs.net 0.86
2. coolwaterdogs.com 0.81
...
Ranked List of Documents
?

Indexing

Our ranking function needs to access a lot of statistics for potentially *every* document. How do we do this efficiently?

- Structure our data such that we can access it in constant $O(1)$ time!

Biggest Challenge

- Term frequency component of BM25 needs the freq of terms in documents
- How do we store counts of terms such that we don’t have to iterate over every document for every query?
- Think about what data structure gives you lookups in constant time

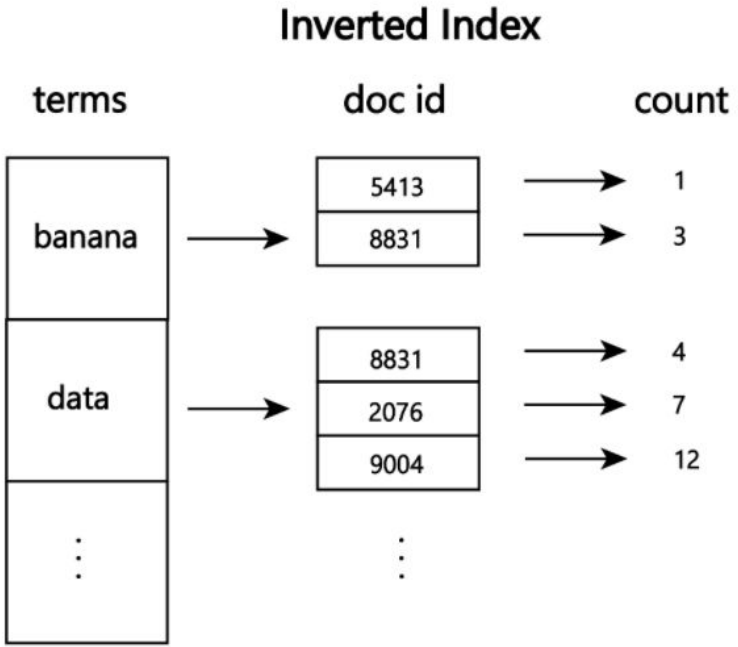
Inverted Index

A hash map.

- term -> documents that term occurs in
- possibly further structure such as frequency of term in that document

When evaluating BM25 for a document/query pair

- Can directly lookup frequency of term in a document



Indexing

What about storing other features?

- Similar idea - use hash maps! (see image)

Challenges at scale:

- Creating and updating this index takes a lot of computation. By indexing, you shift the computational burden away from query-time
- Might not fit all on one server
- Lots of work on compression

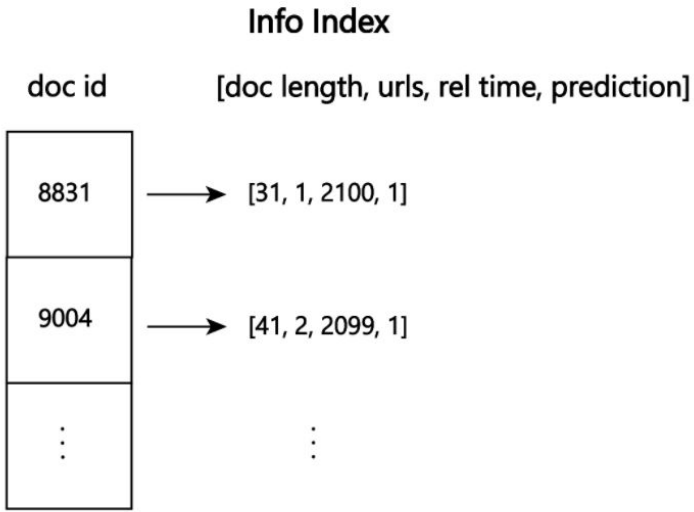
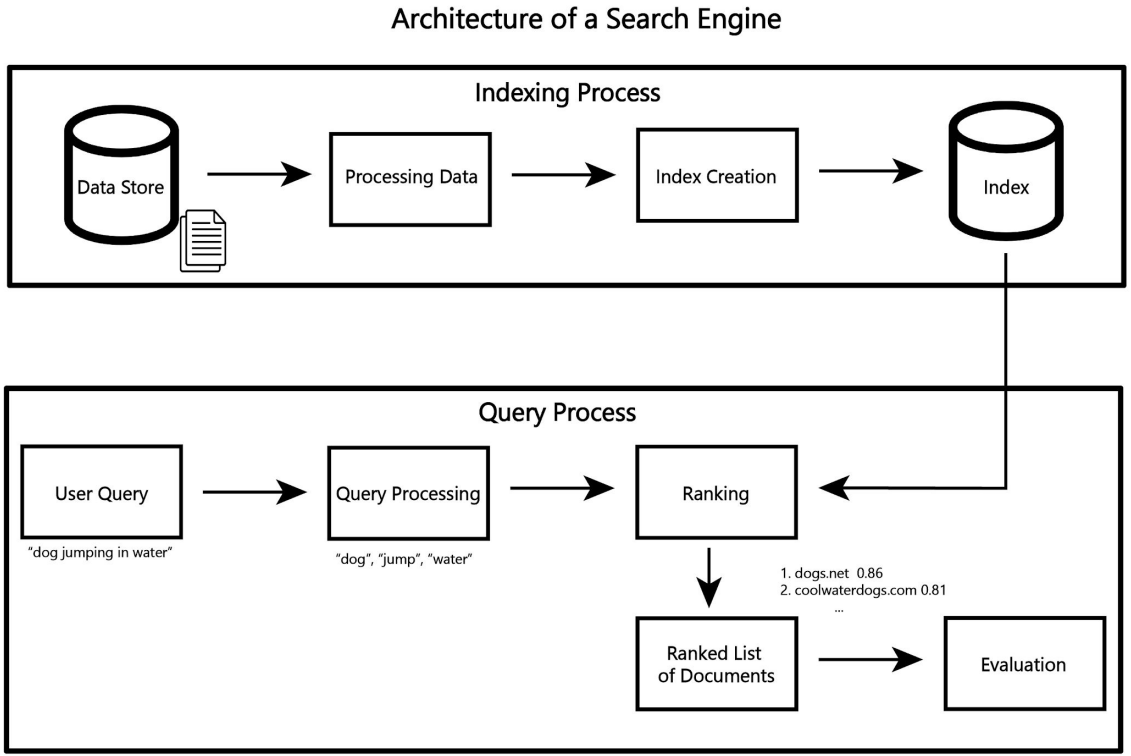
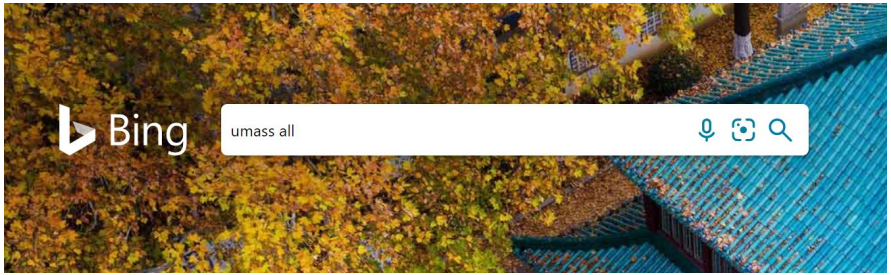


Figure 7: The main data structures in our search engine enabling efficient retrieval.

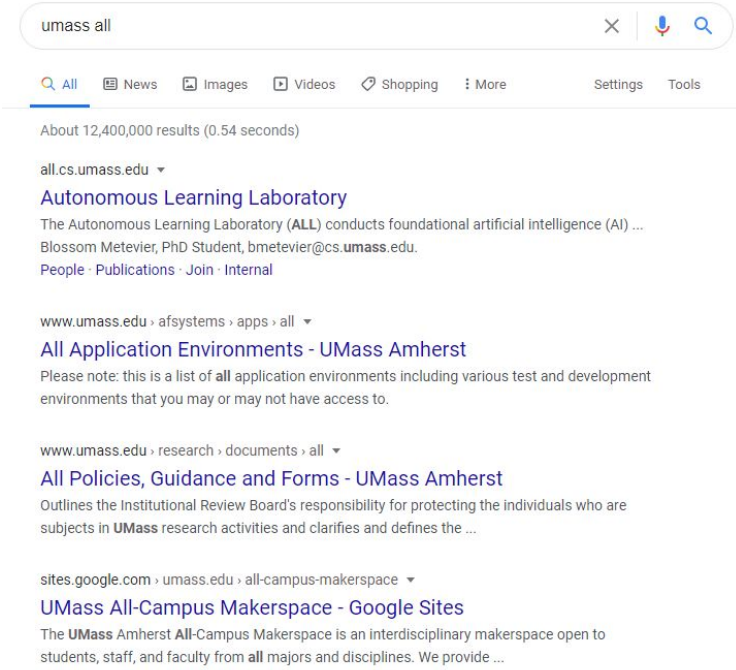
Putting it Together

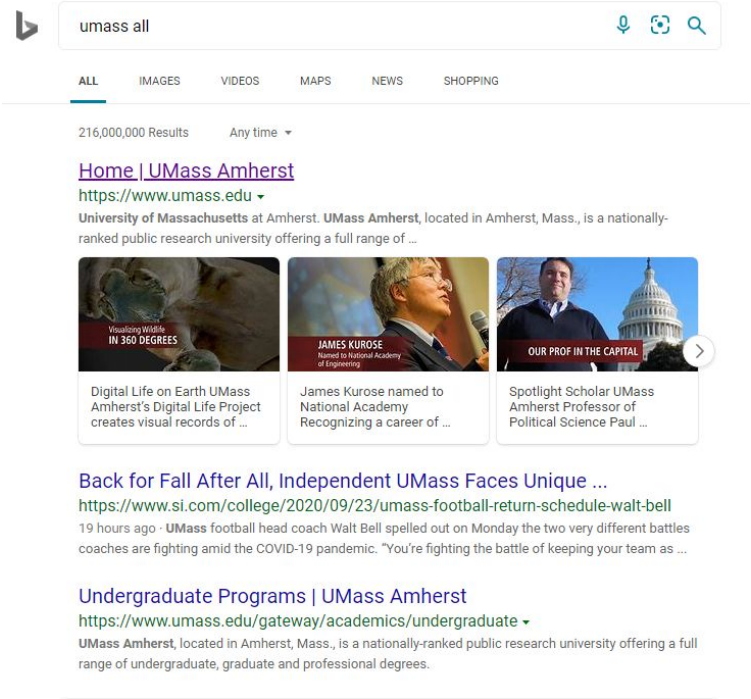


Evaluation



Evaluation


umass all
All
News
Images
Videos
Shopping
More
Settings
Tools
About 12,400,000 results (0.54 seconds)
all.cs.umass.edu
Autonomous Learning Laboratory
The Autonomous Learning Laboratory (ALL) conducts foundational artificial intelligence (AI) ...
Blossom Metevier, PhD Student, bmetevier@cs.umass.edu.
People · Publications · Join · Internal
www.umass.edu › afsystems › apps › all
All Application Environments - UMass Amherst
Please note: this is a list of all application environments including various test and development environments that you may or may not have access to.
www.umass.edu › research › documents › all
All Policies, Guidance and Forms - UMass Amherst
Outlines the Institutional Review Board's responsibility for protecting the individuals who are subjects in UMass research activities and clarifies and defines the ...
sites.google.com › umass.edu › all-campus-makerspace
UMass All-Campus Makerspace - Google Sites
The UMass Amherst All-Campus Makerspace is an interdisciplinary makerspace open to students, staff, and faculty from all majors and disciplines. We provide ...


umass all
ALL
IMAGES
VIDEOS
MAPS
NEWS
SHOPPING
216,000,000 Results
Any time
Home | UMass Amherst
https://www.umass.edu
University of Massachusetts at Amherst. UMass Amherst, located in Amherst, Mass., is a nationally-ranked public research university offering a full range of ...
Visualizing Wildlife IN 360 DEGREES
Digital Life on Earth UMass Amherst's Digital Life Project creates visual records of ...
JAMES KUROSE
Named to National Academy of Engineering
James Kurose named to National Academy Recognizing a career of ...
OUR PROF IN THE CAPITAL
Spotlight Scholar UMass Amherst Professor of Political Science Paul ...
Back for Fall After All, Independent UMass Faces Unique ...
https://www.si.com/college/2020/09/23/umass-football-return-schedule-walt-bell
19 hours ago · UMass football head coach Walt Bell spelled out on Monday the two very different battles coaches are fighting amid the COVID-19 pandemic. "You're fighting the battle of keeping your team as ...
Undergraduate Programs | UMass Amherst
https://www.umass.edu/gateway/academics/undergraduate
UMass Amherst, located in Amherst, Mass., is a nationally-ranked public research university offering a full range of undergraduate, graduate and professional degrees.

Evaluation

Why did Google get it “right”? Data!

When a user searches “umass all” and doesn’t get what they were looking for, what is the first thing they do?

- Immediately after search “umass autonomous learning lab”
- Search engines leverage this information to feed query expansion, spellcheck, etc

Bing’s user base is smaller and a lot of its users are people who use it by default on a Windows PC.

- Likely very few Bing users ever searched for “umass all” and “umass autonomous learning lab” right after

Evaluation

Difficult to determine what search results are relevant

Relevance is subjective and a lot of ambiguity

Conclusion

Gives an idea of how to be a better searcher!

Search Engines combine so many aspects of Computer Science
Even if you don’t ever build a search engine yourself…

- Use the large scale data processing or storage systems that evolved out of building search engines
- Build models for ranking
- NLP
- Leverage data structures to implement efficient systems