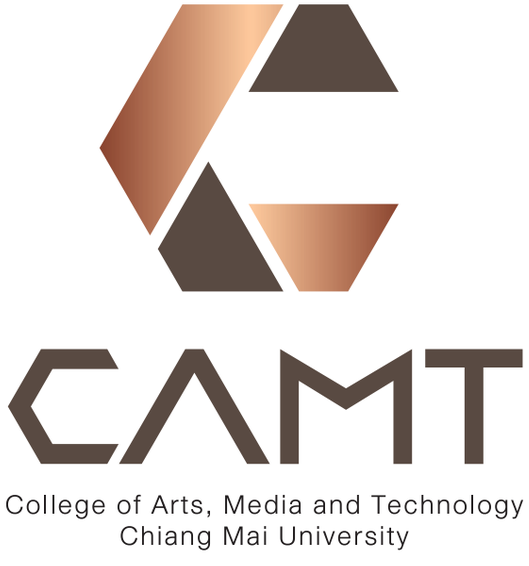
SE 481 Introduction to Information Retrieval (IR for SE)

Module #6 — Web Search Engine



Passakorn Phannachitta, D.Eng.

passakorn.p@cmu.ac.th

College of Arts, Media and Technology

Chiang Mai University, Chiangmai, Thailand

Agenda

- WWW
- Web search engine
- Link analysis

WWW

- Invented by Tim Berners-Lee in 1989 and continued into 1990 at CERN to facilitate information sharing between scientists across the globe.
- Merged the concept of hypertext with the transmission of documents to create **interconnected information** accessible via the Internet.

WWW — Pre history

- Ted Nelson conceptualized **hypertext** in 1960, and he began working on the project that would become Xanadu in 1965.
- Doug Engelbart indeed invented the **mouse** and demonstrated a system that included hypertext features.
- ARPANET, the precursor to the internet was initiated in 1966.
- While foundational technologies like **hypertext** and **networking** were available in the **1970s**, it was the emergence of personal computers and the expansion of global networks in the **following decade** that set the stage for the creation of the web.

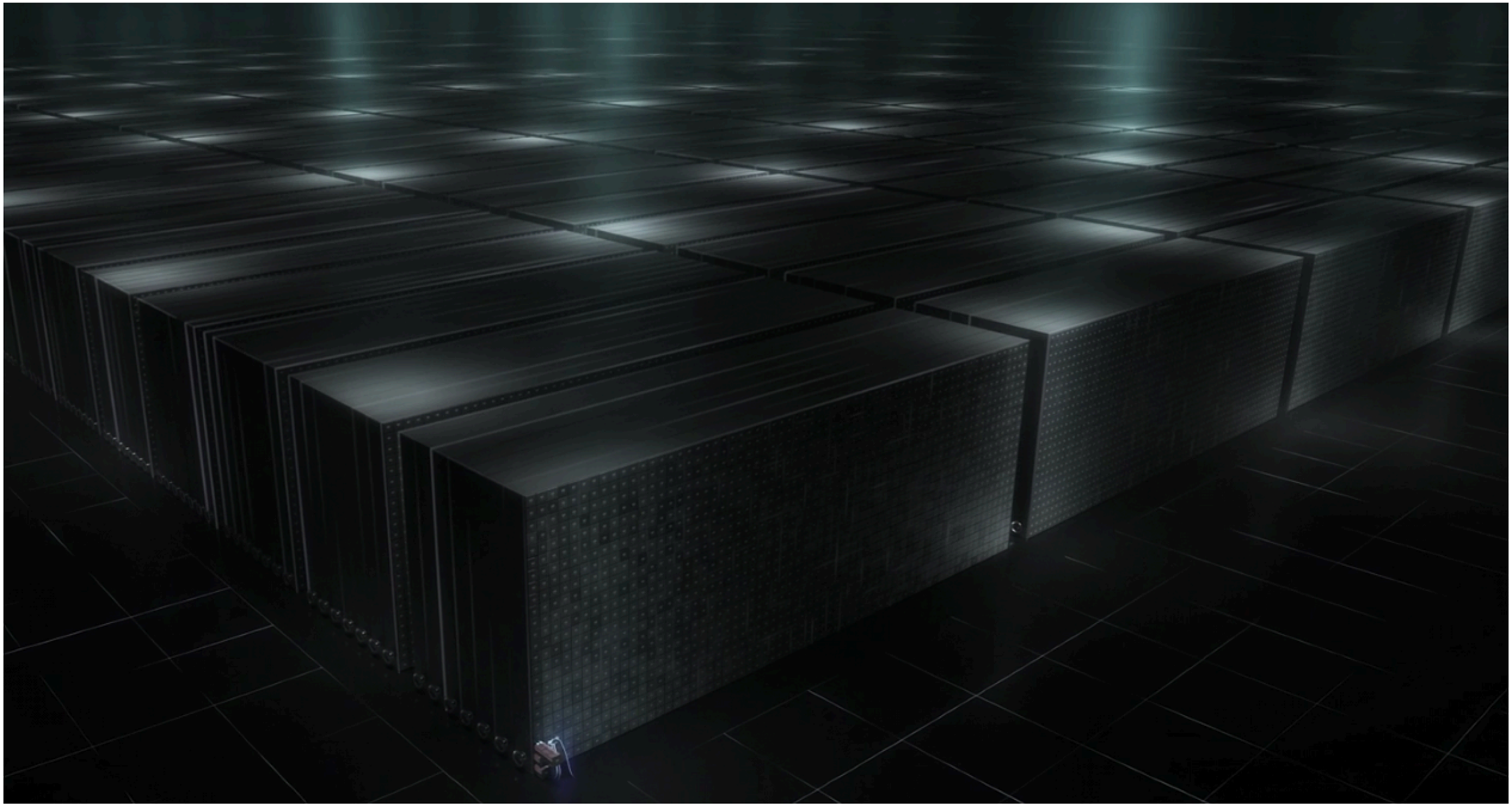
WWW — History

- The earliest web browsers were created around 1991. Notably, a **text-based browser** called Line Mode Browser was developed in 1992.
- Marc Andreessen and Eric Bina developed the Mosaic browser at the University of Illinois at Urbana-Champaign's National Center for Supercomputing Applications (NCSA) in 1993, which significantly contributed to the web's popularity due to its **graphical interface**.

WWW — History

- Marc Andreessen co-founded Mosaic Communications Corporation with entrepreneur Jim Clark in 1994. The company was later renamed **Netscape** Communications to avoid legal issues with NCSA over the name Mosaic.
- Microsoft did indeed license Mosaic technology from Spyglass, Inc., which had licensed it from NCSA, and released **Internet Explorer** in 1995.

Without search engines, the web wouldn’t work



Without search engines, the web wouldn’t work

- There would be no incentive to create contents
- E.g., Pinterest

https://pepper.agency/blog/advertising-on-pinterest/

Search engine — Early history

- By the late 1980s, anonymous FTP was a common method for distributing files.
- In 1990, Alan Emtage at McGill University developed **Archie**, a tool for indexing FTP archives.
  - Compiled lists of files from numerous FTP servers.
  - Enabled regular expression searches of file names.

Search engine — History

- In 1993, early web robots (spiders) were built to collect URL's:
  - Wanderer
  - ALIWEB (Archie-Like Index of the WEB)
  - WWW Worm (indexed URL's and titles for regex search)

- In 1994, Stanford grad students David Filo and Jerry Yang started manually collecting popular web sites into a topical hierarchy called Yahoo.

Search engine — History

- In early 1994, Brian Pinkerton at the University of Washington developed **WebCrawler** for a class project. It was notable for being the first search engine to index entire pages and later became part of **Excite** and **AOL**.
- Shortly thereafter, Michael Mauldin (often misspelled as Fuzzy Maudlin) from Carnegie Mellon University created **Lycos**.
  - It was pioneering in using retrieval techniques from the DARPA-funded **Tipster** project.
  - Lycos was notable for indexing a substantial number of web pages for its time.

Search engine — History

- Developed by DEC in late 1995, **Altavista** ran on a powerful cluster of Alpha processors.
  - It was renowned for its fast processing of queries and the ability to handle boolean operators and phrases.

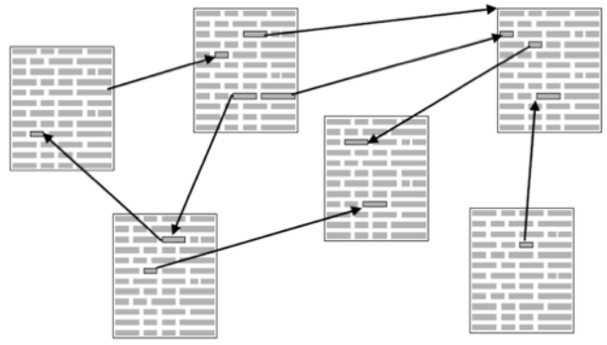
- **Google** founded by Larry Page and Sergey Brin in 1998 during their Ph.D. at Stanford.
  - Innovated with link analysis algorithms like **PageRank** to determine website authority and relevance.

Search engine — History

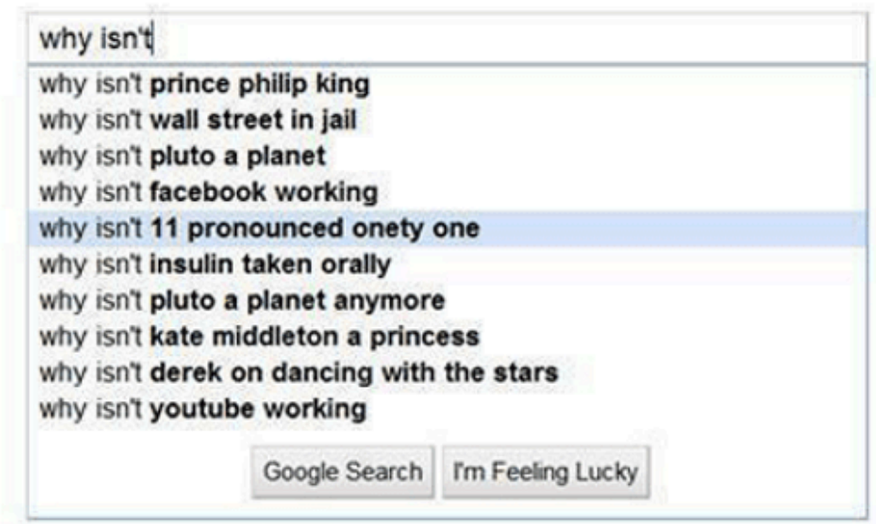
- Microsoft launched MSN Search in 1998 based on **Inktomi** (started from UC Berkeley in 1996), changed to **Live Search** in 2007, and **Bing** in 2009

- **Twitter's Earlybird** introduced around 2012 was designed to maintain a near real-time index of tweets, ensuring that new information was quickly searchable.

Different from traditional IR



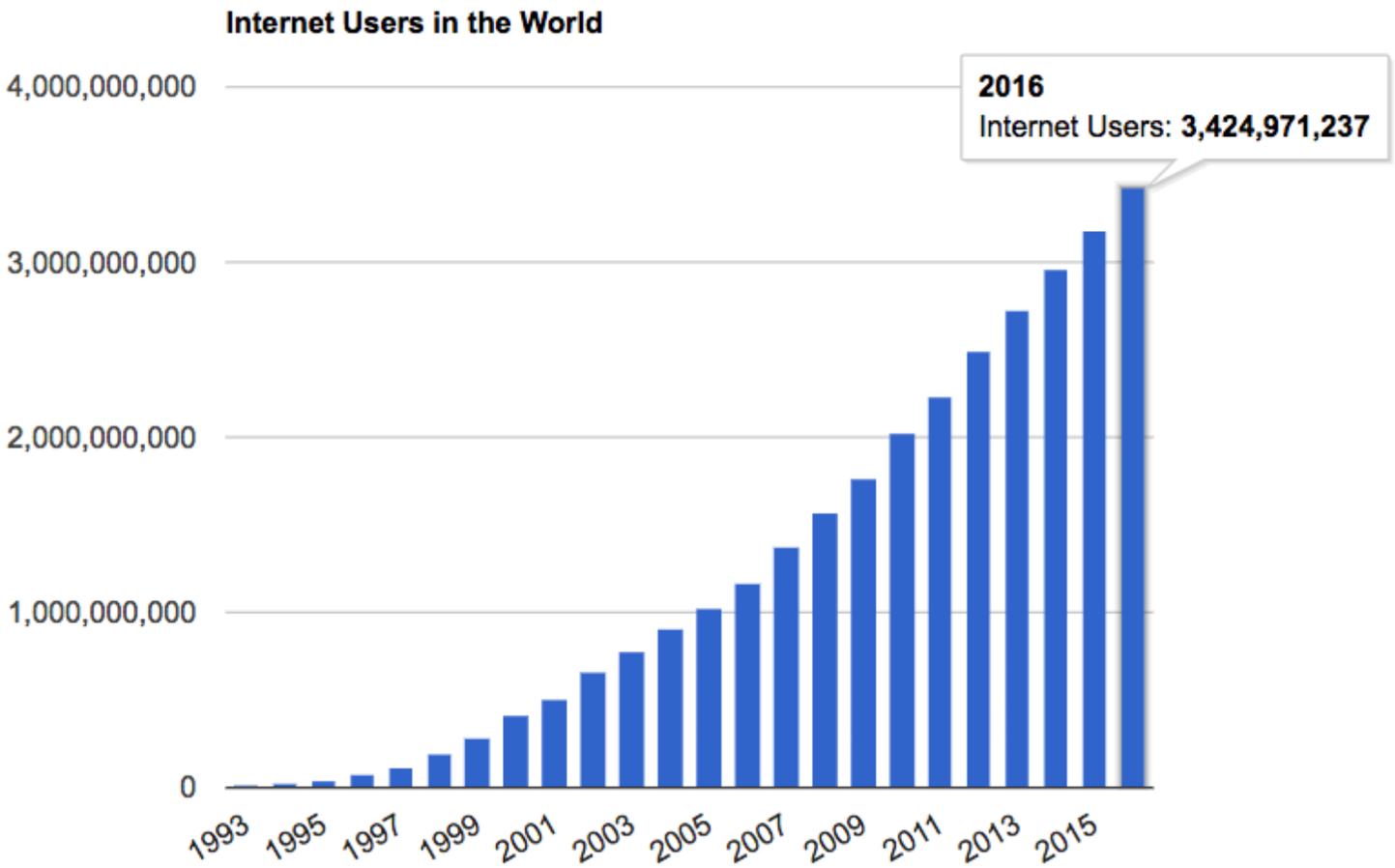
- Web search utilizes the complex network of links between pages.
- There's a vast and varied range of queries, users, and documents, each with unique characteristics.
- Context plays a significant role in web search, influencing the relevance of results.
- The web contains a substantial amount of advertisements and spam that need to be navigated.



Challenges

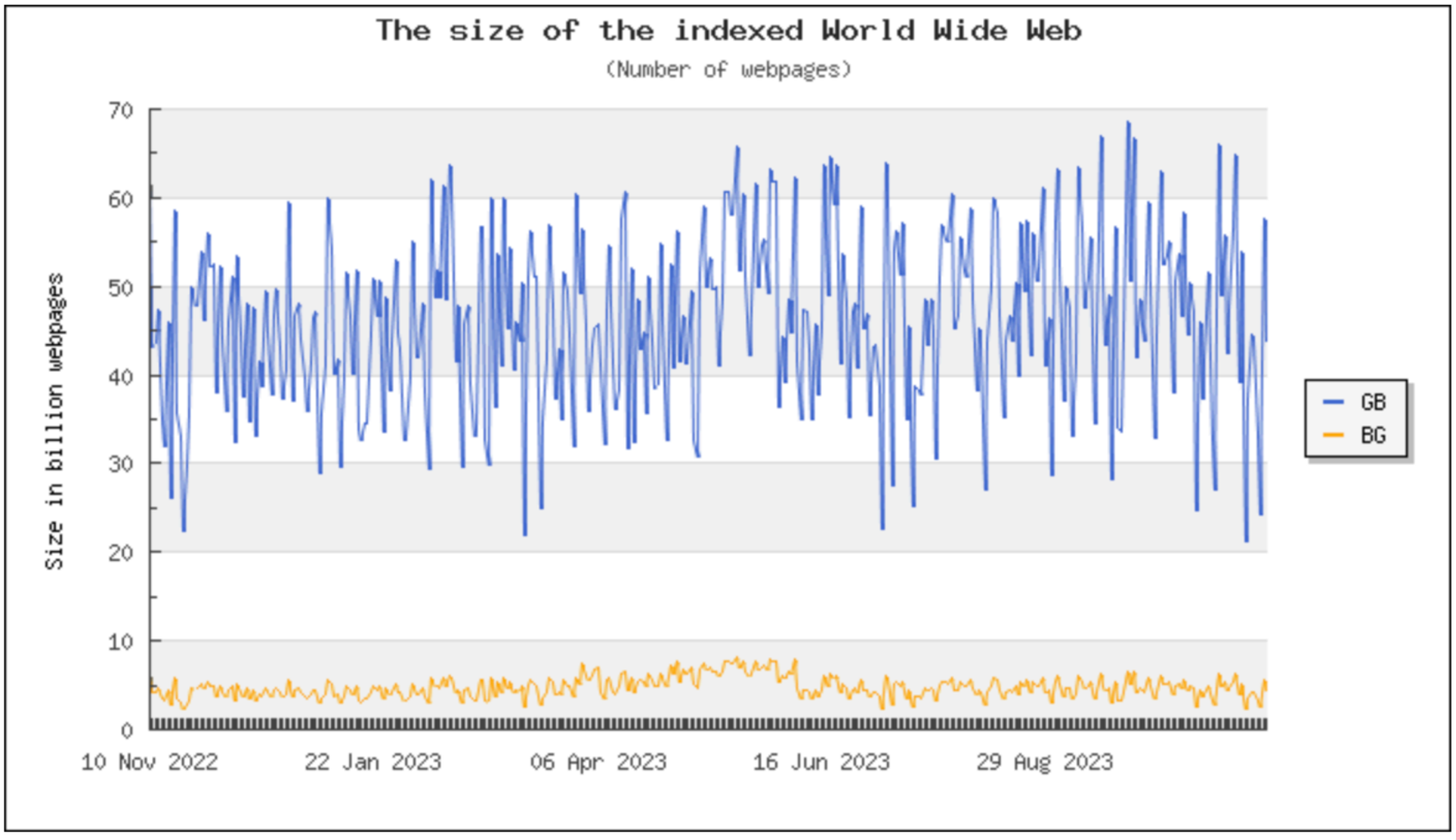
- **Distribution and Volatility** — Documents are dispersed across millions of servers and can frequently change or vanish.
- **Scale and Duplication** — The web comprises billions of documents, including a significant proportion of near-duplicates.
- **Data Quality and Structure** — Varying quality, lack of editorial oversight, and inconsistent HTML structure present unique hurdles.
- **Diversity** — The web features a wide array of media types, languages, and character sets, adding to its complexity.

#Internet users



http://www.internetlivestats.com/internet-users/#trend

Current size of the web



GB = Sorted on Google and Bing

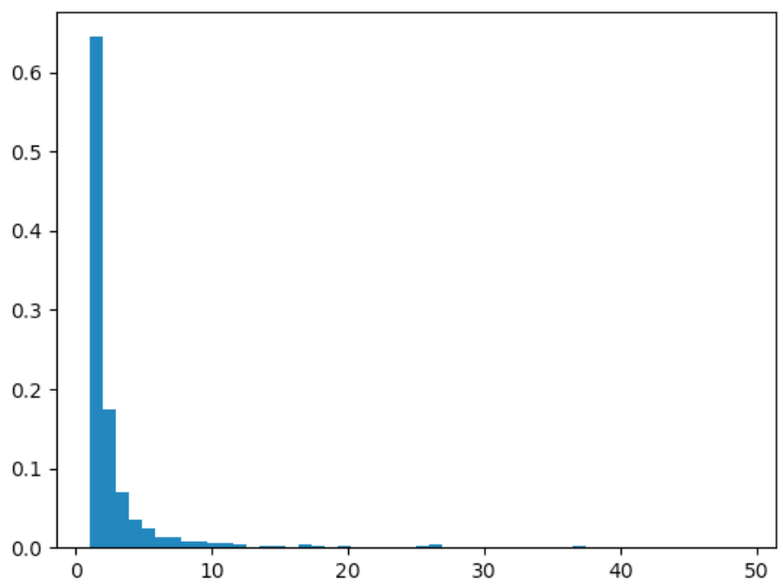
BG = Sorted on Bing and Google

https://www.worldwidewebsize.com/

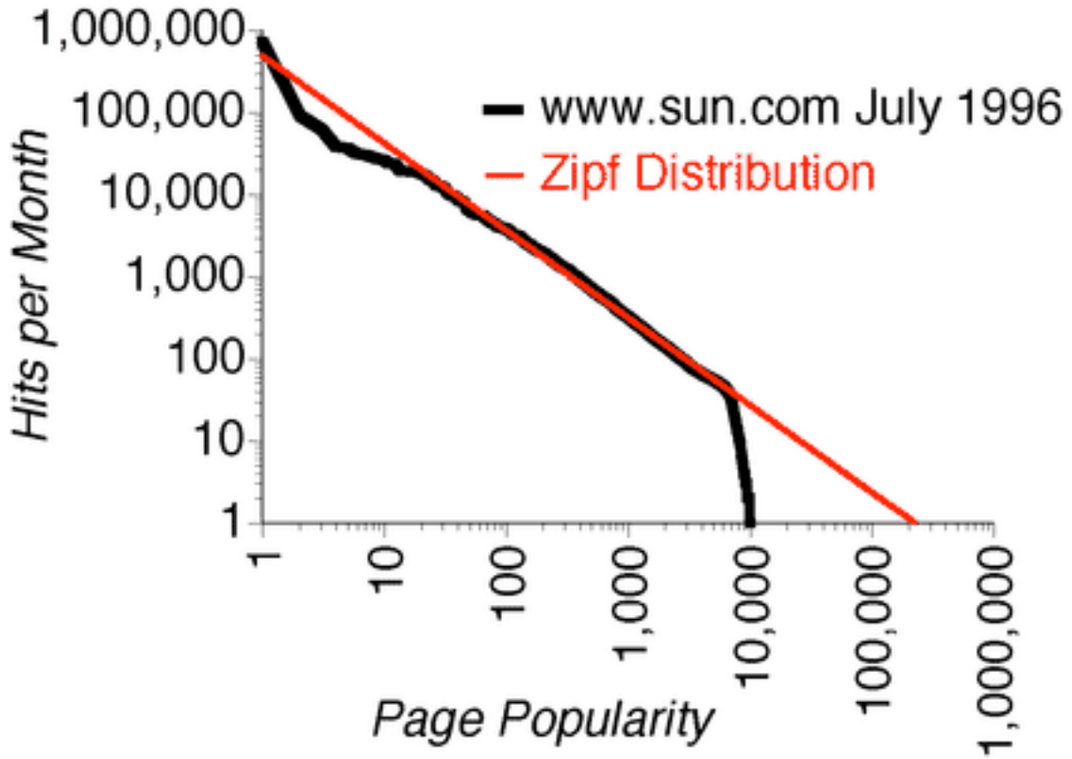
Zipf’s law distribution

- The count of inbound and outbound links to a page often follows a **Zipfian** distribution.
- The distribution of web page lengths also adheres to **Zipf's law**.
- Visits to web pages typically exhibit a **Zipfian** distribution, with a few pages receiving most hits.

$$p(x) = \frac{x^{-}1}{\zeta(a)}$$



Zipf's law and page popularity



https://www.nngroup.com/articles/zipf-curves-and-website-popularity/

Business models for web search

- **Banner ad payments** — Advertisers pay for banner ads on the site that do not depend on a user's query.
  - **CPM**: Cost Per Mille (thousand impressions) — Advertisers pay for every thousand views of their ad.
  - **CPC**: Cost Per Click — Payment occurs only when an ad is clicked.
  - **CTR**: Click Through Rate — The ratio of clicks to ad views. CPC = CPM / (CTR * 1000)
  - **CPA**: Cost Per Action (Acquisition) — Costs are incurred only when an ad click results in a purchase..
- Advertisers bid for "keywords". Ads for highest bidders displayed when user query contains a purchased keyword.
  - **PPC**: Pay Per Click. CPC for bid word ads (e.g. Google AdWords).

Business models for web search

- **Keyword bidding** — Advertisers place bids on keywords, with the highest bidders' ads showing for relevant user queries.
  - **PPC** (Pay Per Click) — A form of CPC applied to keyword ads, such as Google AdWords.

Business models for web search — History

- Banner ads with **CPM** payment dominated early web advertising.
- GoTo Inc., established in 1997, pioneered the **PPC** bidding model and obtained a patent for it.
- Google launched AdWords with a PPC model in the fall of 2000.
- GoTo rebranded as Overture in October 2001.
- Overture filed a lawsuit against Google over **PPC** usage in April 2002.
- Yahoo acquired Overture in October 2003.
- Google settled with Overture/Yahoo, granting 2.7 million shares of Class A common stock in August 2004.

Where's Google™ making its money?

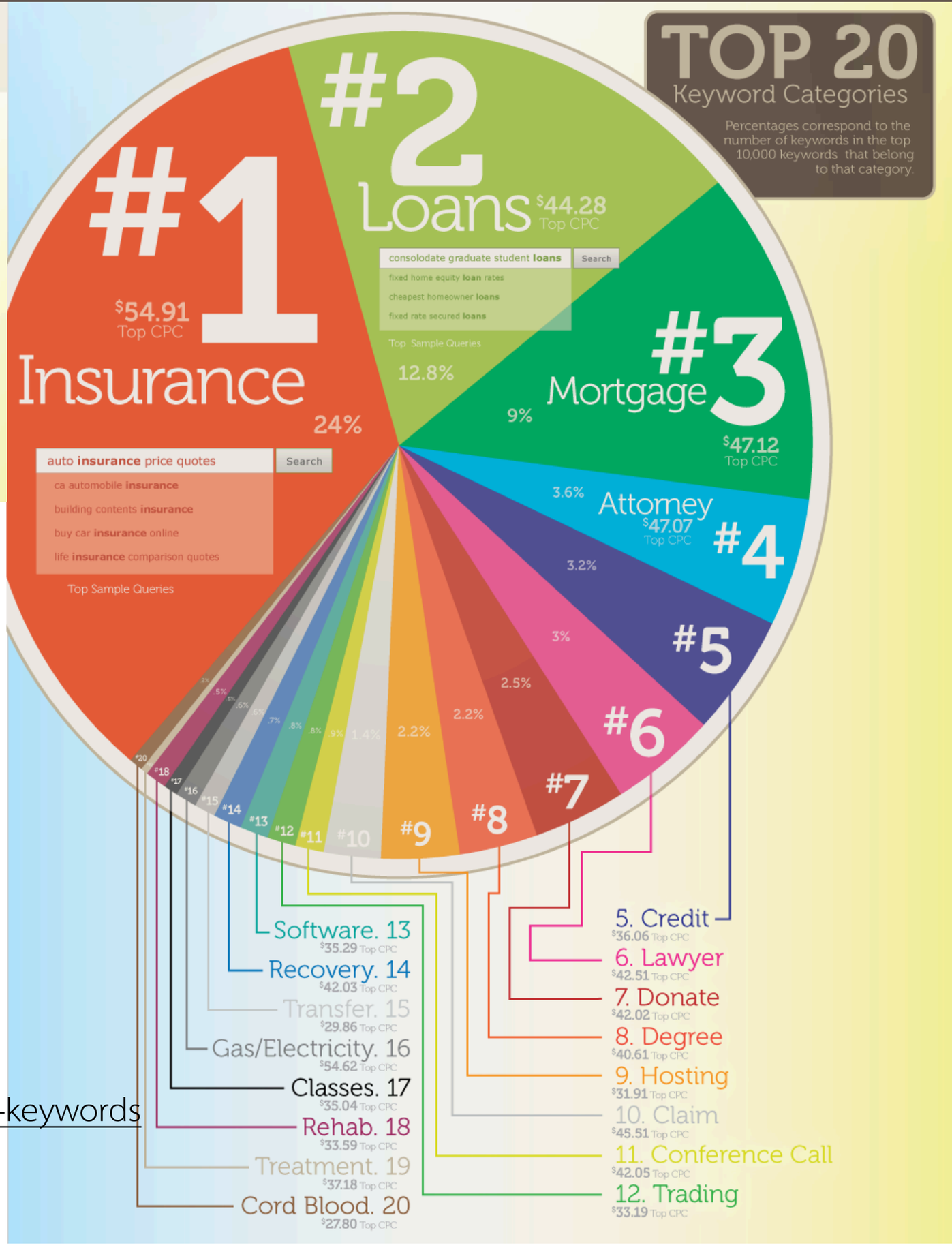
$32.2 billion
in total advertising revenue

SPOILER ALERT: ADVERTISING!

$33.3 billion
in total revenue *in Q3 2010 - Q2 2011*

of Google's Revenue
97%
is from advertising

Top 20
Most Expensive Keywords
in Google™ AdWords Advertising



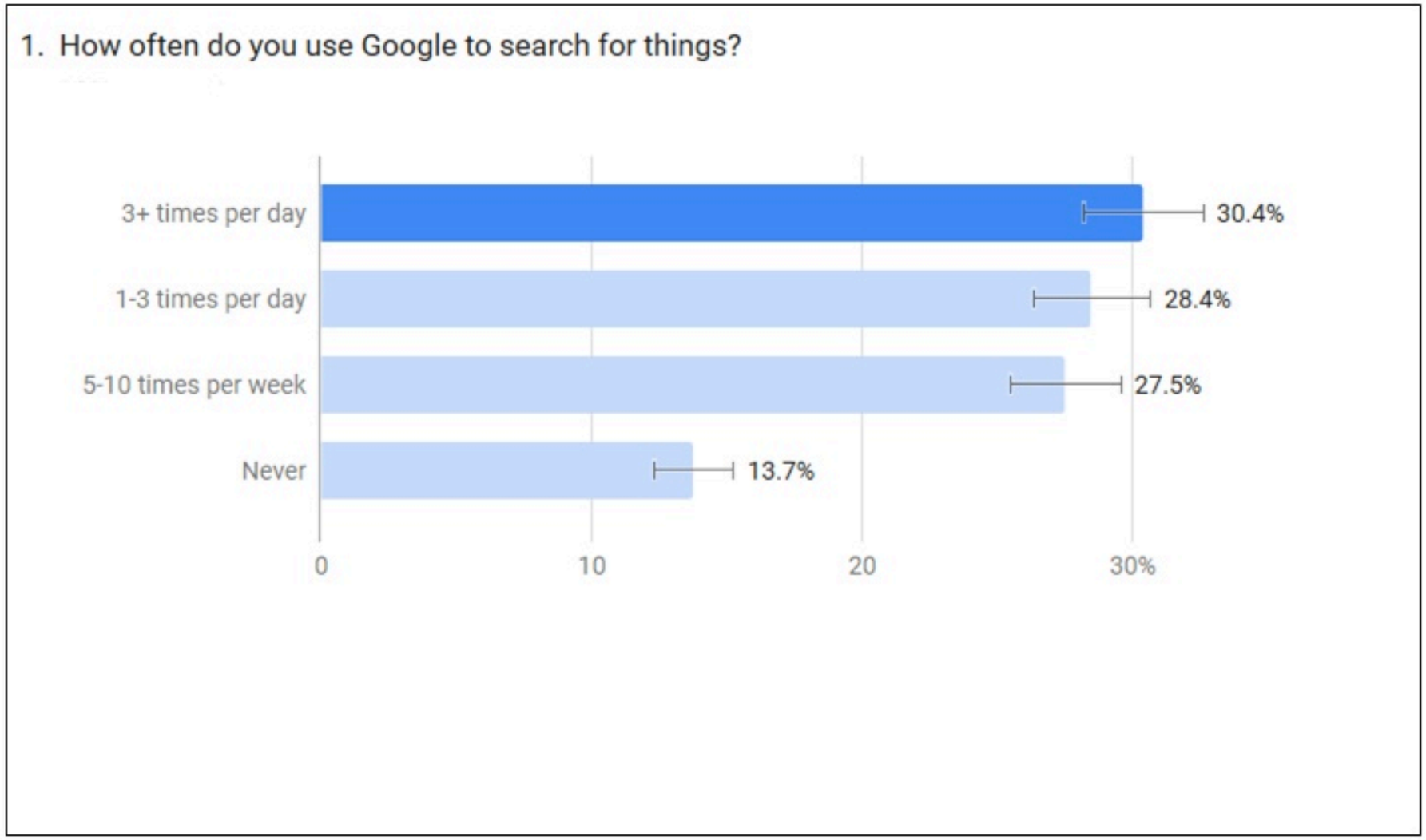
https://www.wordstream.com/articles/most-expensive-keywords

CAMT 2023
College of Arts, Media and Technology
Chiang Mai University

presented by WordStream

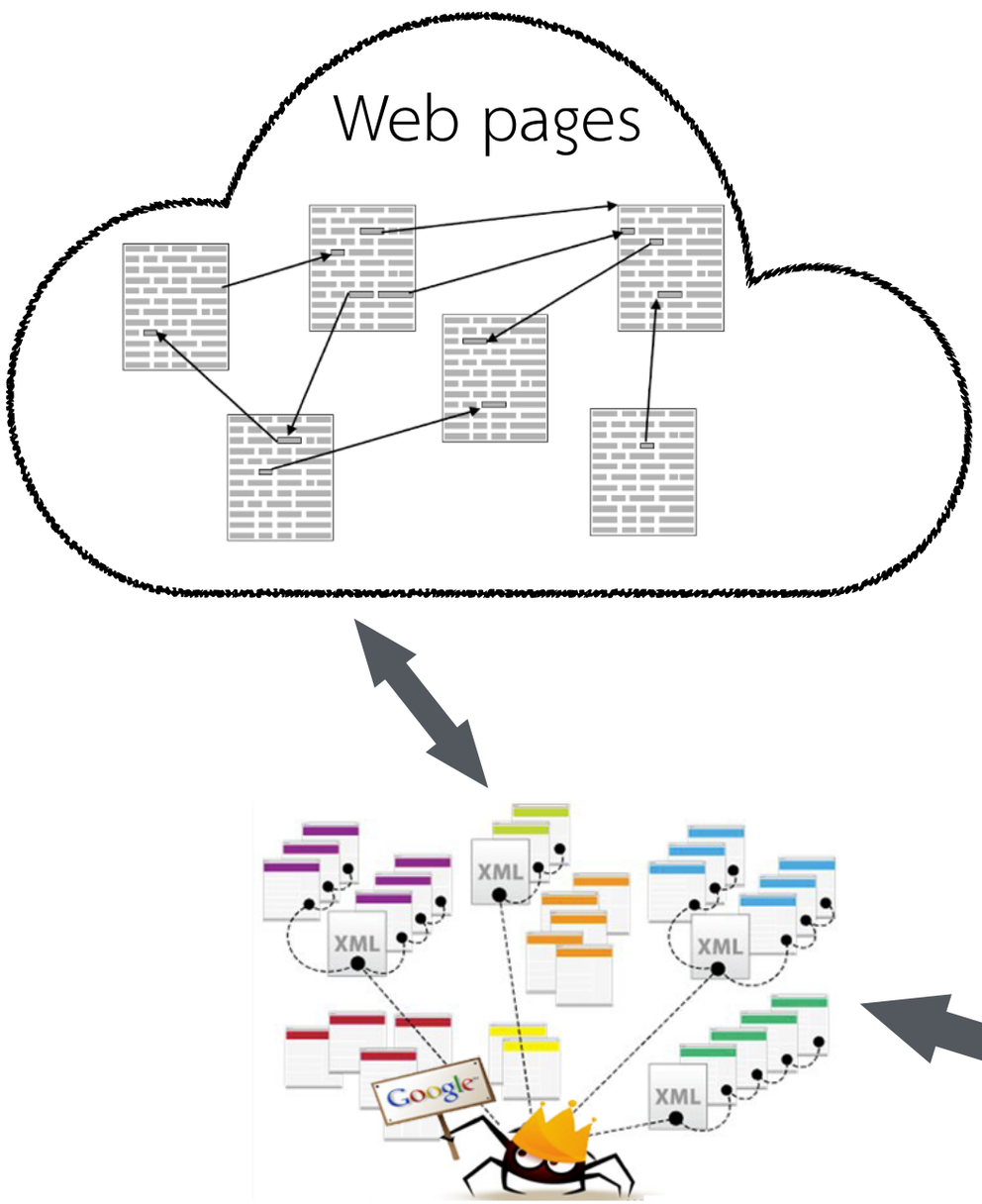
developed by: nowsourcing.com

According to Teknicks

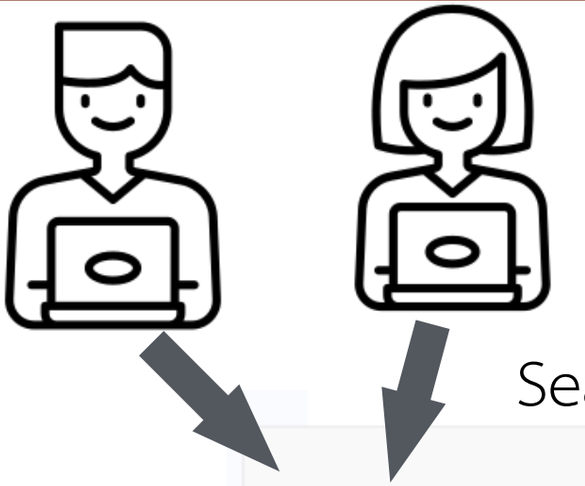


teknicks™

Web search using IR



Web crawler



Search interface

Indexer

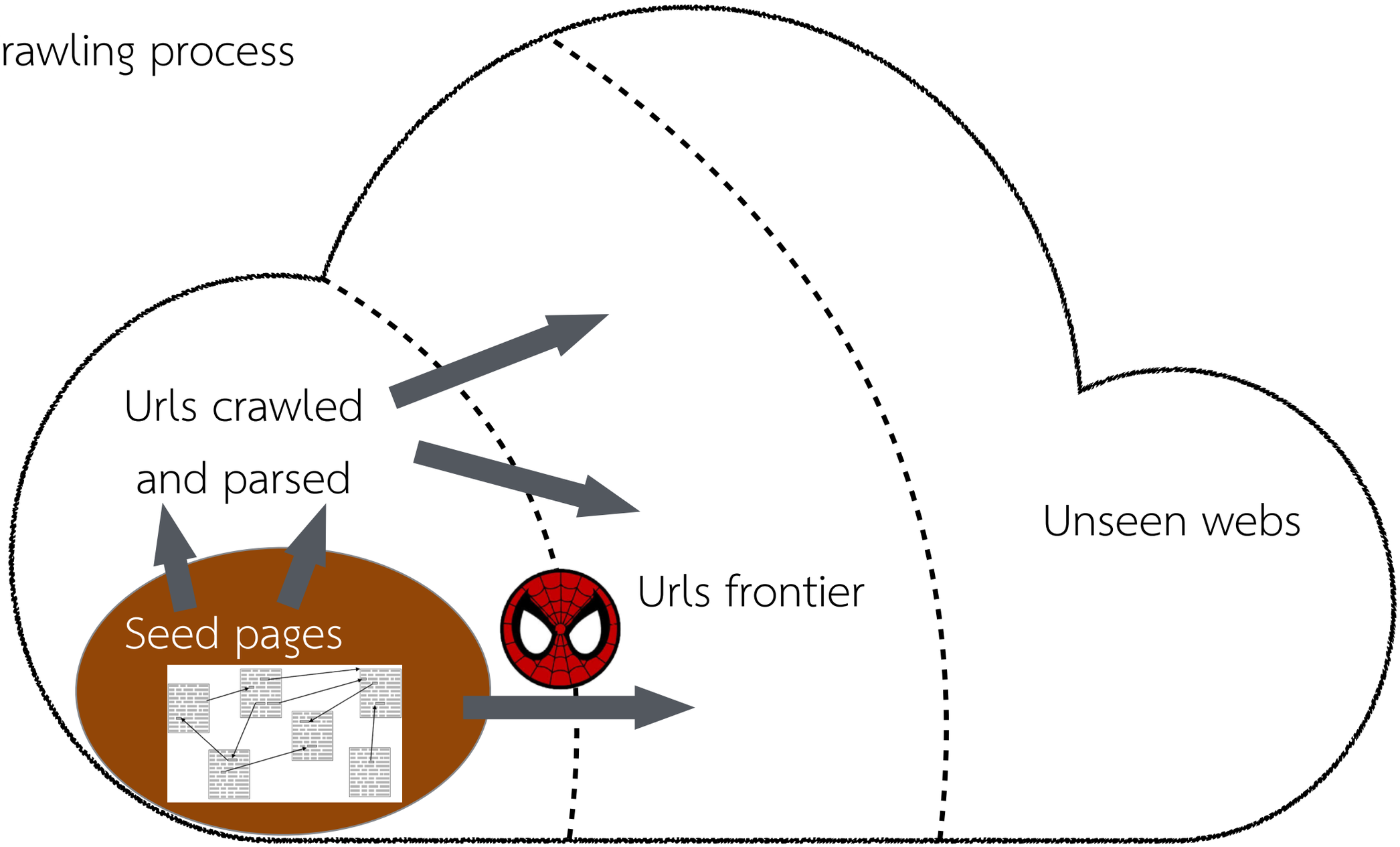
Indexes

Web crawler

- **Also known as** — Web spiders, robots, bots, crawlers.
- Crawling Procedure:
  - **Initialization** — Start with a list of seed URLs placed in a queue.
  - **Processing loop**
    - Fetch each URL's content.
    - Parse the page to extract new URLs.
    - Recursively queue new URLs for crawling.
  - **Tracking** — Maintain a lookup table of processed URLs and a queue for those pending processing.
  - **Iteration** — Continue the cycle with new URLs from the queue.

Graphical explanation

Crawling process



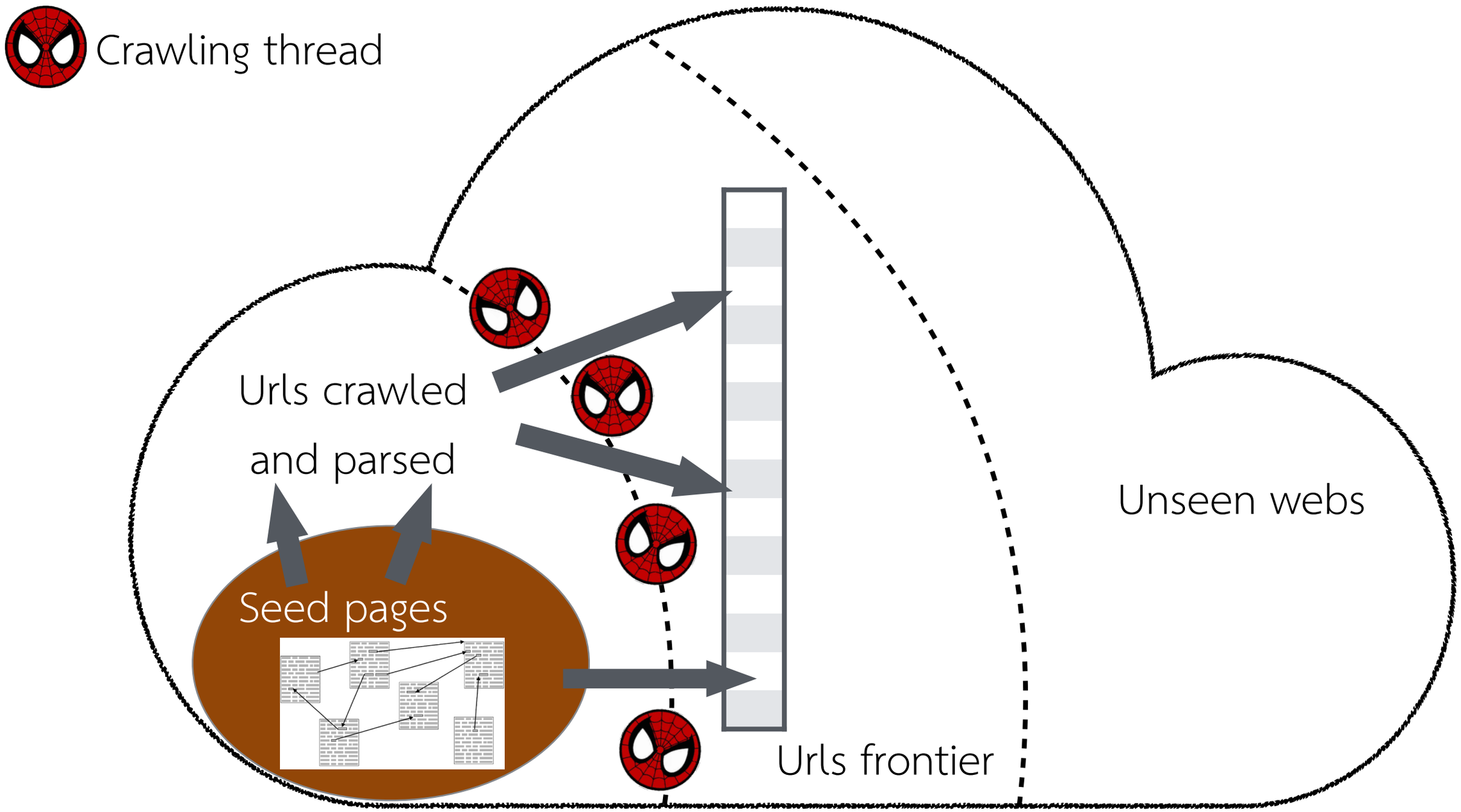
Some difficulties

- The scale of the web often exceeds the capacity of a single machine, necessitating distributed computing.
- Managing the constraints of latency and bandwidth during data retrieval.
- Navigating through and identifying malicious pages.
- Dealing with mirrors and duplicate content.
- Maintaining web etiquette.

What to concern when building a crawler

- Capability
  - Designed for parallel processing.
  - Scalable to handle the web's vastness.
  - Efficient in resource usage and execution.
- Robustness
  - Resilient against various forms of malicious web content and server behaviors.
- Politeness
  - Adheres to both the stated and unstated rules of web etiquette, including respecting robots.txt files and server load.

Scalability



Duplication avoidance ~ efficiency

- URL normalization
  - E.g., https://www.jetbrains.com/pycharm/#chooseYourEdition

https://www.jetbrains.com/pycharm

- Utilize a large hash table to store and check normalized URLs, employing tools like **memCached** or **Redis**.
- Plus implementing more sophisticated algorithms, such as:
  - Network algorithms like TCP's slow start for rate control.
  - Statistical methods like Markov chains for prediction modeling.
  - Machine learning approaches, including neural networks, for intelligent decision-making.

Crawler etiquette

- Politeness principle
  - Adhere to both overt and subtle rules of web etiquette.
- Explicit Politeness
  - Comply with webmaster directives on site accessibility.
    - Observe rules set forth in robots.txt files.
- Implicit Politeness
  - Exercise restraint in the absence of guidelines to prevent overloading servers.

Robots.txt

- It is an access control protocol guiding web crawlers on permitted areas of a website.
- Established in 1994 to communicate site crawling preferences to web robots.
- Websites specify crawl permissions in a /robots.txt file at the root directory.
  - Details elaborated in http://www.robotstxt.org/robotstxt.html
  - E.g.,

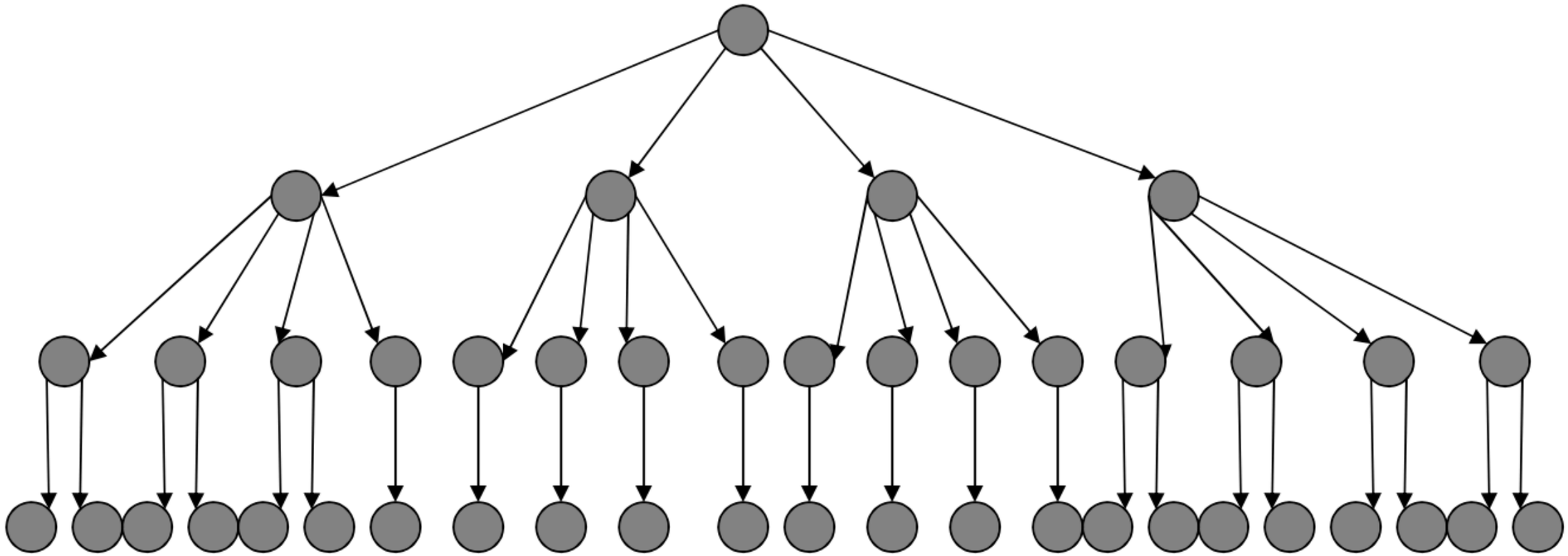
```
User-agent: *
Disallow: /yoursite/temp/

User-agent: searchengine
Disallow:
```

Prevents robots from accessing specified directories, like "/yoursite/temp/", with exceptions for specified agents.

Graph-search algorithm

- Breadth-first search (BFS) or Depth-first search (DFS)?



Link extraction

- E.g., extract all links from https://cmu.ac.th/en/faculty/aboutus
  - 1 levels
  - 2 levels

- Note — Python has Scrappy but it does not fully support multithreading, so let’s try building one.

A simple multithreaded web crawler

```
01  class MultiThreadCrawler:
02      def __init__(self, base_url, depth):
03          self.base_url = base_url
04          extracted_url = urlparse(base_url)
05          parent = extracted_url.path[:extracted_url.path.rfind("/") + 1]
06          self.root_url = '{}://{}{}'.format(extracted_url.scheme, extracted_url.netloc, parent)
07          self.pool = ThreadPoolExecutor(max_workers=multiprocessing.cpu_count() - 1)
08          self.to_crawl = Queue()
09          self.to_crawl.put({self.base_url: depth})
10          self.stored_folder = Path(os.path.abspath('')).parent / 'crawled/'
11
12          if not Path(self.stored_folder).exists():
13              Path.mkdir(self.stored_folder)
14
15          if Path(self.stored_folder / 'url_list.pickle').exists():
16              with open(self.stored_folder / 'url_list.pickle', 'rb') as f:
17                  self.crawled_pages = pickle.load(f)
18              print(self.crawled_pages)
19          else:
20              self.crawled_pages = set([])
```

Entry point

Current directory

Create a queue

Add the entry url to the queue

Continue from the saved work or just start a new one

A simple multithreaded web crawler

```
1   def extract_page(self, obj):
2       if obj.result():
3           result, url, depth = obj.result()
4           if result and result.status_code == 200:
5               url_lists = self.parse_links(result.text, depth)
6               self.parse_contents(url, result.text, url_lists)
```

```
1   def get_page(self, url, depth):
2       try:
3           res = requests.get(url, timeout=(3, 30)) ◀.............. Download the webpage
4           return res, url, depth
5       except requests.RequestException:
6           return
```

A simple multithreaded web crawler

```
01  def parse_links(self, html, depth):
02      soup = BeautifulSoup(html, 'html.parser')
03      links = soup.find_all('a', href=True)
04      url_lists = []
05      for link in links:
06          url = link['href']
07          url = urljoin(self.root_url, url)
08          if depth >= 0 and '..' not in url and url not in self.crawled_pages:
09              print("Adding {}".format(url))
10              self.to_crawl.put({url: depth})
11              url_lists.append(url)
12      return url_lists
```

Find all links by extracting all the <a href>

Make full path from relative path if not already a full path

Add to the queue if not crawled depth has not reached the threshold.

A simple multithreaded web crawler

```
01  def parse_contents(self, url, html, url_lists):
02      def tag_visible(element):
03          if element.parent.name in ['style', 'script', 'head', 'title', 'meta', '[document]']:
04              return False
05          if isinstance(element, Comment):
06              return False
07          return True
08
09      try:
10          soup = BeautifulSoup(html, 'html.parser')
11          texts = soup.findAll(string=True)
12          visible_texts = filter(tag_visible, texts)
13
14          title = soup.find('title').string.strip()
15          text = u" ".join(t.strip() for t in visible_texts).strip()
16
17          with open(self.stored_folder / (str(hash(url)) + '.txt'), 'w', encoding='utf-8') as f:
18              json.dump({'url': url, 'title': title, 'text': text, 'url_lists': url_lists}, f,
ensure_ascii=False)
19      except:
20          pass
```

Get all the texts from visible tags

Parse into json format and write to a textual file

A simple multithreaded web crawler

```
01      def run_scraper(self):
02          while True:
03              try:
04                  target = self.to_crawl.get(timeout=10)
05                  url, depth = [(k, target[k]) for k in target][0]
06                  if url not in self.crawled_pages:
07                      self.crawled_pages.add(url)
08                      job = self.pool.submit(self.get_page, url, depth - 1)
09                      job.add_done_callback(self.extract_page)
10              except Empty:
11                  with open(self.stored_folder / 'url_list.pickle', 'wb') as f:
12                      pickle.dump(self.crawled_pages, f, pickle.HIGHEST_PROTOCOL)
13                  with open(self.stored_folder / 'url_list.pickle', 'rb') as f:
14                      print(pickle.load(f))
15                  break
16              except Exception as e:
17                  print(e)
18                  continue
19
20
21  if __name__ == '__main__':
22      s = MultiThreadCrawler("https://camt.cmu.ac.th/index.php/en/", 2)
23      s.run_scraper()
```

Dequeue the first entry

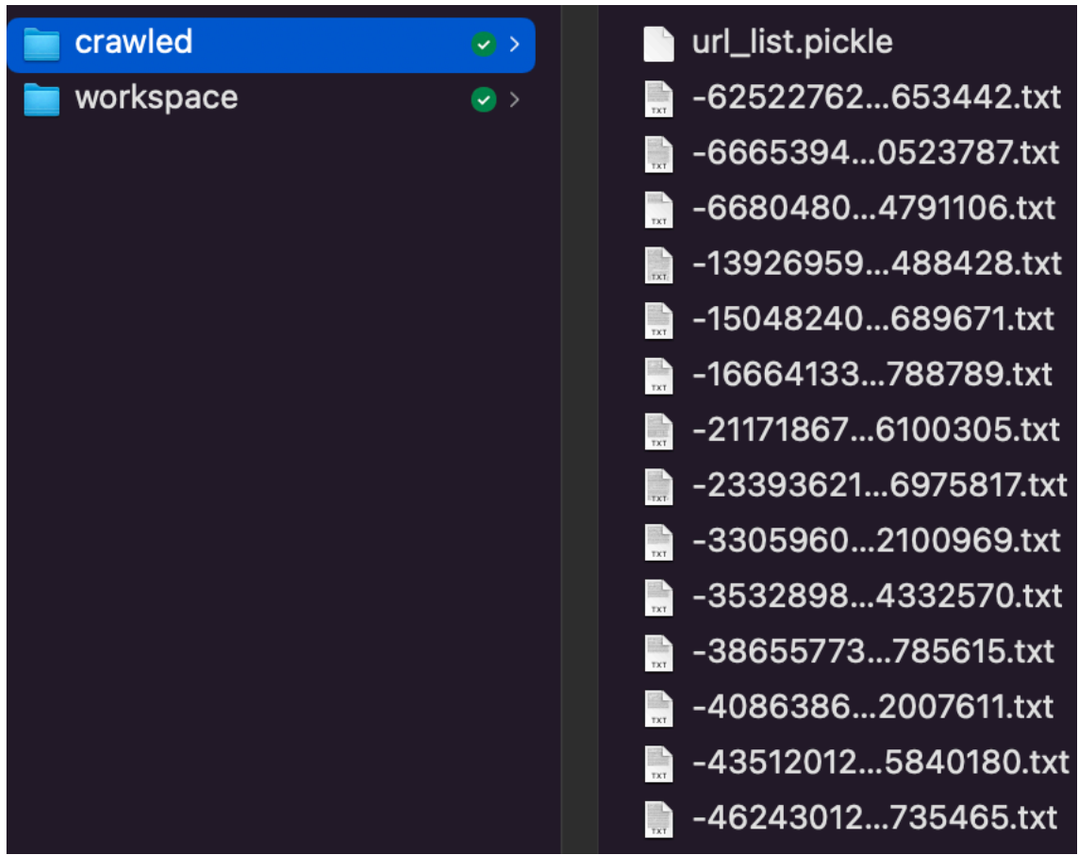
If it has never been processed

If we cannot dequeue,
then it means our task is done

An entry url example

A simple multithreaded web crawler

- Output of execution
  - A binary file (pickle) containing a mapping from webpage URLs to their hashed values, utilized as filenames.
  - A collection of JSON files, each representing a webpage with fields like {title: , text:, }



A simple multithreaded web crawler

- Ideas for some enhancements
  - Transition from file-based storage to database systems for improved organization and retrieval.
  - Implement content checks, such as language detection, prior to storage to ensure relevance and quality.

A simple web indexer

The crawled folder

```
01  def __init__(self):
02      self.crawled_folder = Path(os.path.abspath('')).parent / 'crawled/'
03      self.stored_file = 'src/resource/manual_indexer.pkl'
04      if os.path.isfile(self.stored_file):
05          with open(self.stored_file, 'rb') as f:
06              cached_dict = pickle.load(f)
07          self.__dict__.update(cached_dict)
08      else:
09          self.run_indexer()
10
11  def run_indexer(self):
12      documents = []
13      for file in os.listdir(self.crawled_folder):
14          if file.endswith(".txt"):
15              j = json.load(open(os.path.join(self.crawled_folder, file)))
16              documents.append(j)
17      self.documents = pd.DataFrame.from_dict(documents)
18      tfidf_vectorizor = TfidfVectorizer(preprocessor=preProcess,
stop_words=stopwords.words('english'))
19      self.bm25 = BM25(tfidf_vectorizor)
20      self.bm25.fit(self.documents.apply(lambda s: ' '.join(s[['title', 'text']]), axis=1))
21      with open(self.stored_file, 'wb') as f:
22          pickle.dump(self.__dict__, f)
```

Load each document into memory, preprocess, transform into *tf-idf, apply bm25, and store the product back to manual_indexer.pkl*

Quick workout #1

- Search using a query ‘school’ with BM25.
  - You have to retrieve the score list and manage the ranking manually.
  - E.g.,

url	title	text	url_lists	score
https://service.camt.cmu.ac.th/gifted	Gift School 2023	<< คลิกที่นี่ >> ระบบรับสมัคร Gifted School | ...	[https://service.camt.cmu.ac.th/gifted/gifted/...	4.459200
:ps://www.grad.cmu.ac.th/index.php?lang=en	Graduate School, Chiang Mai University	MIdS : Multidisciplinary and Interdisciplinary...	[https://cmu.to/admission/, https://w3.grad.cm...	4.276232
https://www.grad.cmu.ac.th/	Graduate School, Chiang Mai University	MIdS : Multidisciplinary and Interdisciplinary...	[https://cmu.to/admission/, https://w3.grad.cm...	4.276232
:tps://service.camt.cmu.ac.th/gifted/gifted/i...	Gift School 2566	Login Form เลขบัตรประชาชน 13 หลัก ดู P...	[]	4.207181
s://go.camt.cmu.ac.th/index.php/th/major/b...	บูรณาการอุตสาหกรรม	Choose your language ไทย English (UK)...	[http://www.go-camt.com/index.php/th/,	3.085978

Search engine database

Rank			DBMS	Database Model	Score		
Nov 2023	Oct 2023	Nov 2022			Nov 2023	Oct 2023	Nov 2022
1.	1.	1.	Elasticsearch	Search engine, Multi-model	139.62	+2.48	-10.70
2.	2.	2.	Splunk	Search engine	97.32	+4.95	+3.10
3.	3.	3.	Solr	Search engine, Multi-model	44.62	-0.74	-6.71
4.	4.	4.	OpenSearch	Search engine, Multi-model	13.31	+0.39	+3.01
5.	5.	5.	MarkLogic	Multi-model	8.17	-0.03	-1.59
6.	6.	6.	Algolia	Search engine	6.84	-0.48	-0.99
7.	7.	7.	Microsoft Azure Search	Search engine	6.43	-0.06	-0.82
8.	8.	8.	Sphinx	Search engine	5.99	-0.02	-0.88
9.	9.	9.	Virtuoso	Multi-model	5.62	+0.19	-0.22
10.	10.	10.	ArangoDB	Multi-model	4.54	+0.27	-1.30

Ref: https://db-engines.com/en/ranking/search+engine

Elasticsearch

QUERY & ANALYZE

Ask your data questions of all kinds

Search your way

Elasticsearch lets you perform and combine many types of searches — structured, unstructured, geo, metric — any way you want. Start simple with one question and see where it takes you.

Analyze at scale

It's one thing to find the 10 best documents to match your query. But how do you make sense of, say, a billion log lines? Elasticsearch aggregations let you zoom out to explore trends and patterns in your data.

SPEED

Elasticsearch is fast. Really, really fast.

Rapid results

When you get answers instantly, your relationship with your data changes. You can afford to iterate and cover more ground.

Powerful design

Being this fast isn't easy. We've implemented inverted indices with finite state transducers for full-text querying, BKD trees for storing numeric and geo data, and a column store for analytics.

All-inclusive

And since everything is indexed, you're never left with index envy. You can leverage and access all of your data at ludicrously awesome speeds.

Ref: https://www.elastic.co/elasticsearch/

Elasticsearch

- Installation
  - https://www.elastic.co/guide/en/elasticsearch/reference/current/install-elasticsearch.html
  - The simplest way is to use Docker (Install Docker desktop first) — https://www.elastic.co/guide/en/elasticsearch/reference/8.12/docker.html

- Run elasticsearch
  - docker run --name es01 --net elastic -p 9200:9200 -it -m 1GB -e "discovery.type=single-node" docker.elastic.co/elasticsearch/elasticsearch:8.12.0

Elasticsearch

- Following the guideline #5, #6, and #7, you should be able to connect to connect with your Elasticsearch docker

```
(base) kong@mba ~ % curl --cacert http_ca.crt -u elastic:"7L4*ufX=xVOj7qa9LDj=" https://localhost:9200

{
  "name" : "01ee726f679d",
  "cluster_name" : "docker-cluster",
  "cluster_uuid" : "AT0Dfx_PRIuv8xmAr1rPdA",
  "version" : {
    "number" : "8.11.0",
    "build_flavor" : "default",
    "build_type" : "docker",
    "build_hash" : "d9ec3fa628c7b0ba3d25692e277ba26814820b20",
    "build_date" : "2023-11-04T10:04:57.184859352Z",
    "build_snapshot" : false,
    "lucene_version" : "9.8.0",
    "minimum_wire_compatibility_version" : "7.17.0",
    "minimum_index_compatibility_version" : "7.0.0"
  },
  "tagline" : "You Know, for Search"
}
```

Try to connect using python console, e.g.,

```
1   from elasticsearch import Elasticsearch
2
3   es = Elasticsearch("http://localhost:9200", basic_auth=("elastic", "7L4*ufX=xVOj7qa9LDj="),
ca_certs="~/http_ca.crt")
4   es.info().body
```

Don’t forget to `pip install elasticsearch`

Path to the certificated
(Generated in #7)

The generated password
(Generated in #6)

A simple indexer (using elasticsearch)

```
01  def __init__(self):
02      self.crawled_folder = Path(os.path.abspath('')).parent / '../crawled/'
03      with open(self.crawled_folder / 'url_list.pickle', 'rb') as f:
04          self.file_mapper = pickle.load(f)
05      self.es_client = Elasticsearch("http://localhost:9200", basic_auth=("elastic",
"7L4*ufX=xVOj7qa9LDj="), ca_certs="~/http_ca.crt"))
```

The folder storing the crawled data

Connecting to the elasticsearch client,
Change the password to yours.

```
01  def run_indexer(self):
02      self.es_client.options(ignore_status=400).indices.create(index='simple')
03      self.es_client.options(ignore_status=[400, 404]).indices.delete(index='simple')
04
05      for file in os.listdir(self.crawled_folder):
06          if file.endswith(".txt"):
07              j = json.load(open(os.path.join(self.crawled_folder, file)))
08              j['id'] = j['url']
09              print(j)
10              self.es_client.index(index='simple', document=j)
```

For each file, passing it to elasticsearch

A simple indexer (using elasticsearch)

A query example searching the text field

```
1   if __name__ == '__main__':
2       s = Indexer()
3       s.run_indexer()
4       query = {'bool': {'must': [{'match': {'text': 'camt'}}]}}
5       results = s.es_client.search(index='simple', query=query)
6       print("Got %d Hits:" % results['hits']['total']['value'])
7       for hit in results['hits']['hits']:
8           print("The title is '{0} ({1})'.".format(hit["_source"]['title'], hit["_source"]['url']))
9
```

```
4       query = {'bool': {'must': [{'match': {'text': 'examination'}}]}}
```

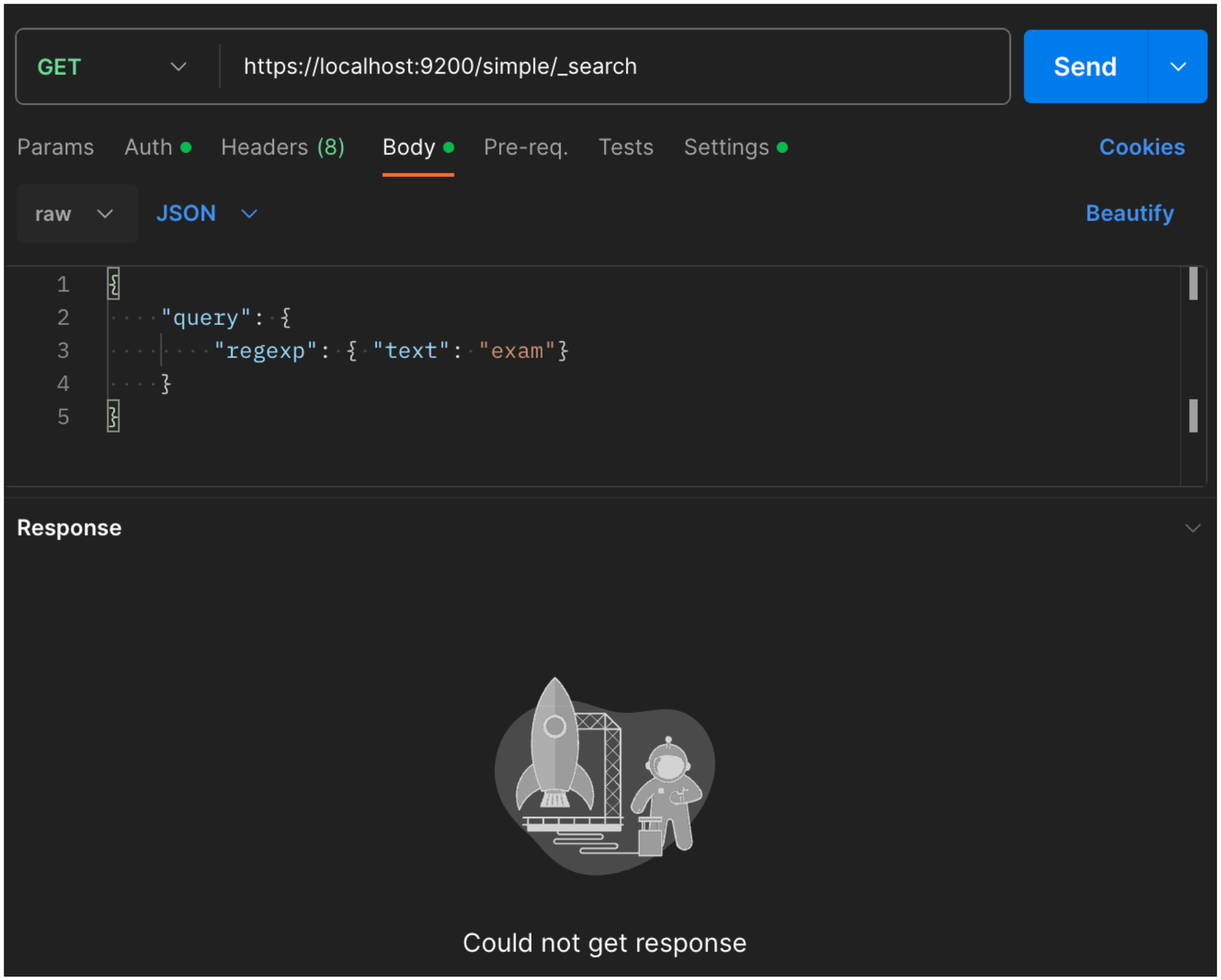
```
4       query = {"regexp": { "text": ".*vision.*"}}
```

```
4       query = {"match": { "text": "vision"}}
```

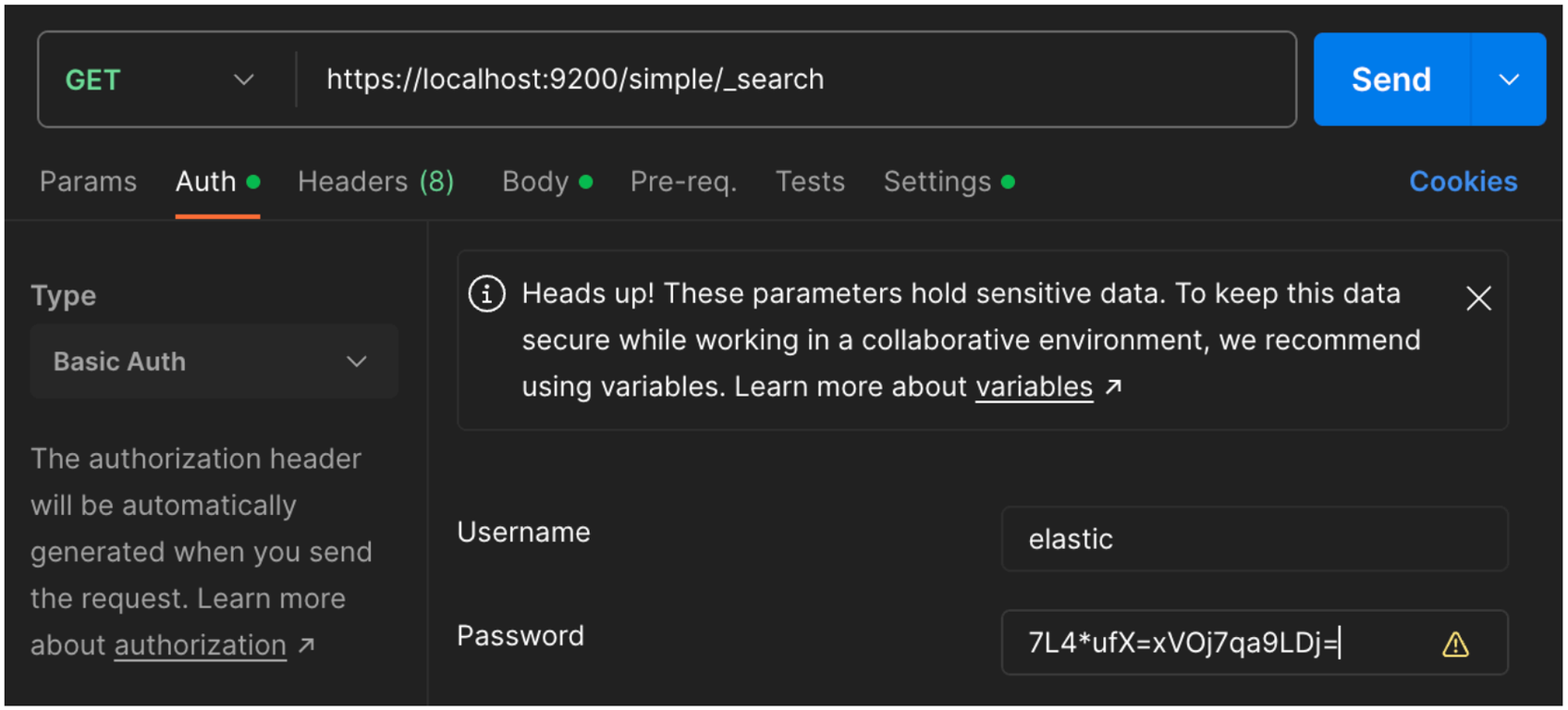
Search result example

```
Got 43 Hits:
The title is 'ข่าวประกาศจากวิทยาลัยฯ (https://camt.cmu.ac.th/index.php/th/หัวข้อกลุ่มข่าวทั้งหมด/54-ข่าวประกาศจากวิทยาลัยฯ.html)'.
The title is 'ตารางสอบปลายภาค 1/2566 (เพิ่มเติม)  (Final Examination 1/2023) (https://camt.cmu.ac.th/index.php/th/หัวข้อกลุ่มข่าวทั้งหมด/54-ข่าวประกาศจากวิทยาลัยฯ/997-final_exam_166_1.html)'.
The title is 'CAMT Announcement (https://camt.cmu.ac.th/index.php/en/all-news-groups/54-camt-announcement.html)'.
The title is 'ตารางสอบปลายภาค 1/2566 (เพิ่มเติม)  (Final Examination 1/2023) (https://camt.cmu.ac.th/index.php/en/all-news-groups/54-camt-announcement/997-final_exam_166_1.html)'.
The title is 'หน้าหลัก (https://www.camt.cmu.ac.th)'.
The title is 'หน้าหลัก (https://camt.cmu.ac.th/)'.
The title is 'ประกาศ (https://camt.cmu.ac.th/index.php/th/2-uncategorised/324-ประกาศ.html)'.
The title is 'หน้าหลัก (https://www.camt.cmu.ac.th/)'.
The title is 'หน้าหลัก (https://camt.cmu.ac.th/index.php/en/?p=&lang=th)'.
The title is 'หน้าหลัก (https://camt.cmu.ac.th/index.php/th/)'.
```

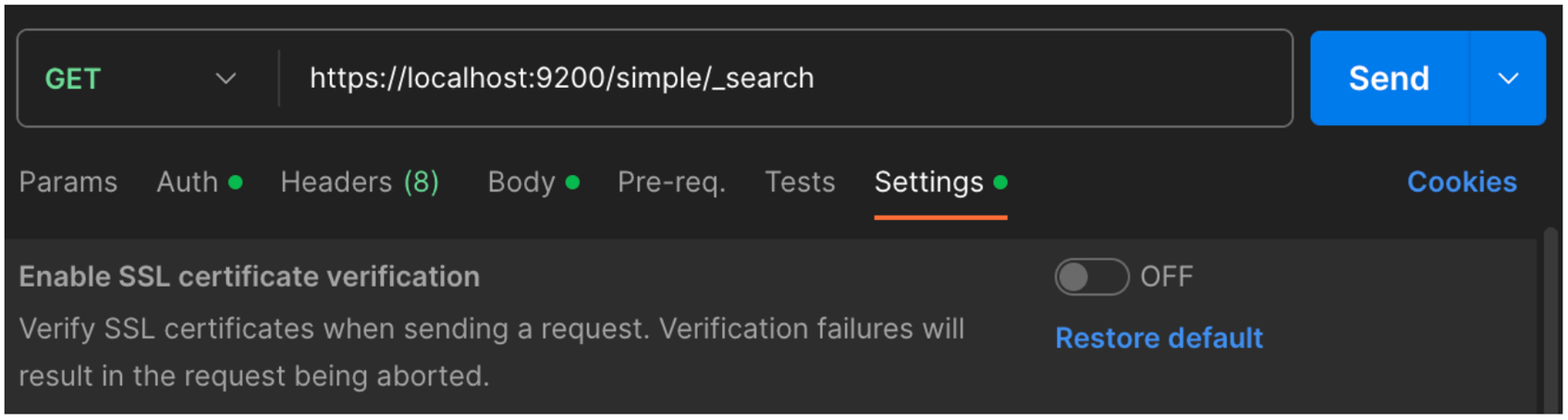
Connect with Postman, e.g.,



Connect with Postman, e.g.,



The generated user and password (Generated in #6)



Turn off SSL verification

Connect with Postman, e.g.,

GET https://localhost:9200/simple/_search Send

Params Auth Headers (8) Body Pre-req. Tests Settings Cookies

raw JSON Beautify

```
1  {
2      "query": {
3          "regexp": { "text": "exam"}
4      }
5  }
```

Body 200 OK 38 ms 10 KB Save as Example

Pretty Raw Preview Visualize JSON

```
23          ],
24          "_source": {
25              "url": "https://www1.reg.cmu.ac.th/reg-ipas/main/index.php",
26              "title": "CMU-IPAS",
27              "text": "Chiang Mai University  CMU     Welcome     Program
                    Information     Announcement     TCAS and CMU-IPAS
                    Admission Timeline     Guideline for Photo/Document
                    How to apply CMU-IPAS     Apply Now
                    Contact            International Program Admission
                    System +       Application for Admission to the
                    International  Programs for the 2023 Academic Year
                    Chiang Mai University is now open for applications for
                    international undergraduate programs for the 2023
                    academic year.    Applicants are required to check
                    requirements of the relevant Annexes of your applied
                    program.   The international programs opened for
                    admission are:   >> Chiang Mai University Business
                    School   •\tBachelor of Accountancy (ACC)   >> College
```

Connect with Postman, e.g.,

SE481_IR / **ES_Local**

Save

GET https://localhost:9200/simple/_search

Send

Params Auth Headers (8) Body Pre-req. Tests Settings Cookies

raw JSON Beautify

```
{
    "_source": {
        "exclude": ["url_lists"]
    },
    "query": {
        "regexp": { "text": ".*exam.*"}
    }
}
```

Body 200 OK 37 ms 108.82 KB Save as Example

Pretty Raw Preview Visualize JSON

```
            "text.keyword"
        ],
        "_source": {
            "url": "https://camt.cmu.ac.th/index.php/en/index.php#about",
            "title": "Home",
            "text": "Home About us Back Vision and
                Mission Map List of CAMT's staff
                organizational structure Programs &
                Admissions Student Back Requesting
                for Recommendation Form System Download documents
                for reimbursement for education fees for children of
                civil servants. Download documents for
                students. Registration for WIL Complaint
```

Adding a data formatting layer

```
01  from flask import Flask, request
...

06  app = Flask(__name__)
07  app.es_client = Elasticsearch("https://localhost:9200", basic_auth=("elastic",
"7L4*ufX=xVOj7qa9LDj="), ca_certs="~/http_ca.crt")
08
09  @app.route('/search_es', methods=['GET'])
10  def search_es():
11      start = time.time()
12      response_object = {'status': 'success'}
13      argList = request.args.to_dict(flat=False)
14      query_term=argList['query'][0]
15      results = app.es_client.search(index='simple', source_excludes=['url_lists'], size=100,
            query={"match": { "text": query_term}})
16      end = time.time()
17      total_hit = results['hits']['total']['value']
18      results_df = pd.DataFrame([[hit["_source"]['title'], hit["_source"]['url'], hit["_source"]
['text'][:100], hit["_score"]] for hit in results['hits']['hits']], columns=['title', 'url', 'text',
'score'])
19
20      response_object['total_hit'] = total_hit
21      response_object['results'] = results_df.to_dict('records')
22      response_object['elapse'] = end - start
23
24      return response_object
25
26  if __name__ == '__main__':
27      app.run(debug=False)
```

Don't forget to `pip install flask`

Modify to yours

Try with Postman or a web browser

localhost:5000/search_es?query=school


```
// 20231111143036
// http://localhost:5000/search_es?query=school

{
  "elapse": 0.10399699211120605,
  "results": [
    {
      "score": 4.850322,
      "text": "Choose your language         ไทย li dir=\"ltr\" class=\"lang-active\"> English (UK)
      "title": "Gifted School 2020",
      "url": "https://go.camt.cmu.ac.th/index.php/th/2019-05-16-09-02-18/2019-05-16-09-05-06"
    },
    {
      "score": 4.7045307,
      "text": "MIdS : Multidisciplinary and Interdisciplinary School   Chiang Mai University   Admission 2024  Grad",
      "title": "Graduate School, Chiang Mai University",
      "url": "https://www.grad.cmu.ac.th/index.php?lang=en"
    },
    {
      "score": 4.7045307,
      "text": "MIdS : Multidisciplinary and Interdisciplinary School   Chiang Mai University   Admission 2024  Grad",
      "title": "Graduate School, Chiang Mai University",
      "url": "https://www.grad.cmu.ac.th/"
    },
    {
      "score": 4.18734,
      "text": "<< คลิกที่นี่ >> ระบบรับสมัคร Gifted School | Pre College",
      "title": "Gift School 2023",
      "url": "https://service.camt.cmu.ac.th/gifted"
    },
    {
      "score": 4.0827875,
      "text": "Login Form  เลขบัตรประชาชน 13 หลัก        ดู Profile         Gifted School 2023  ©2016 All Rights Re",
      "title": "Gift School 2566",
      "url": "https://service.camt.cmu.ac.th/gifted/gifted/index"
    },
```


College of Arts, Media and Technology
Chiang Mai University

Quick exercise

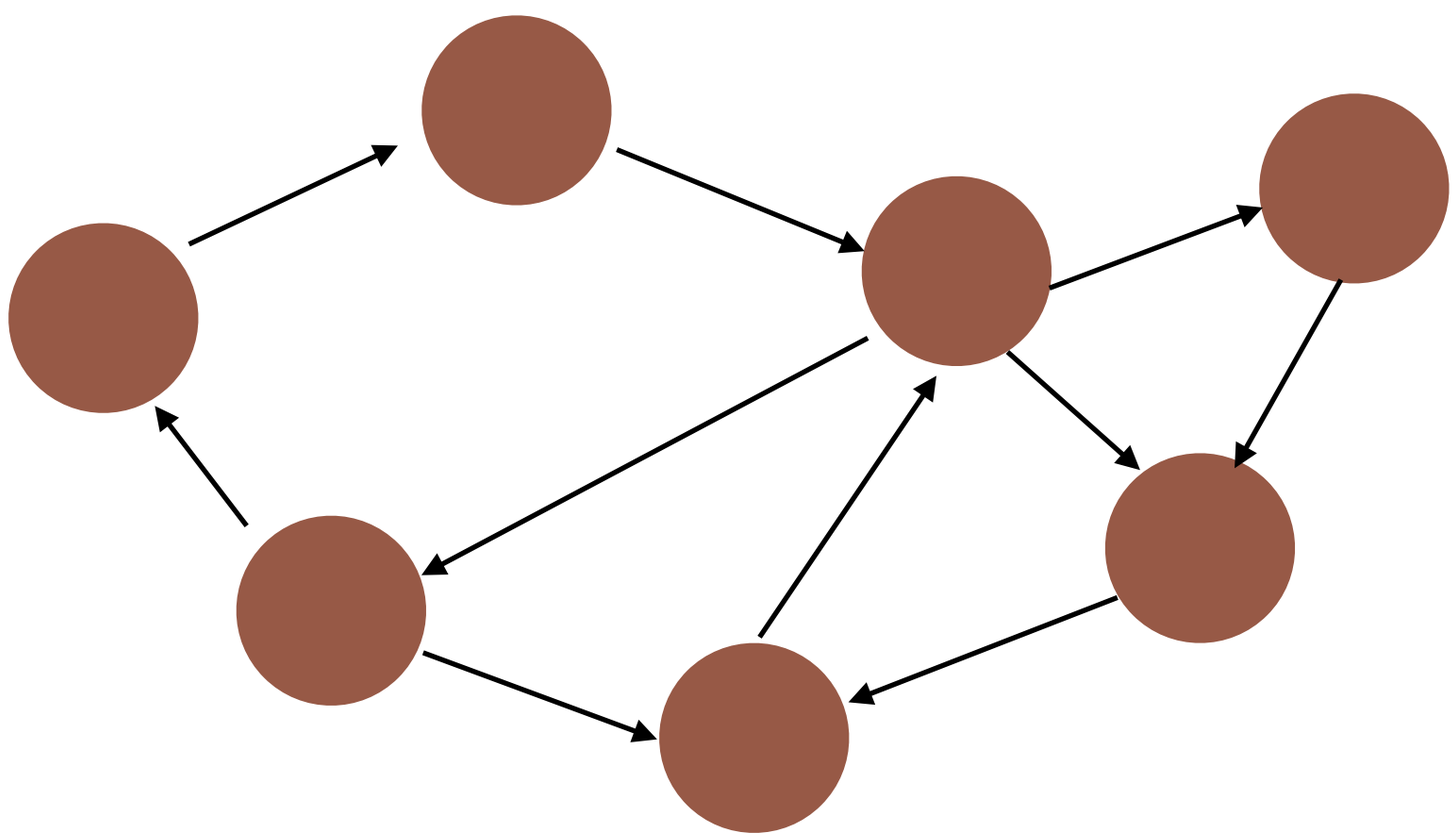
- Modify the flask application to let the user perform searching using **manual_index** with route /search_manual

- Check with postman or web browser

Link analysis

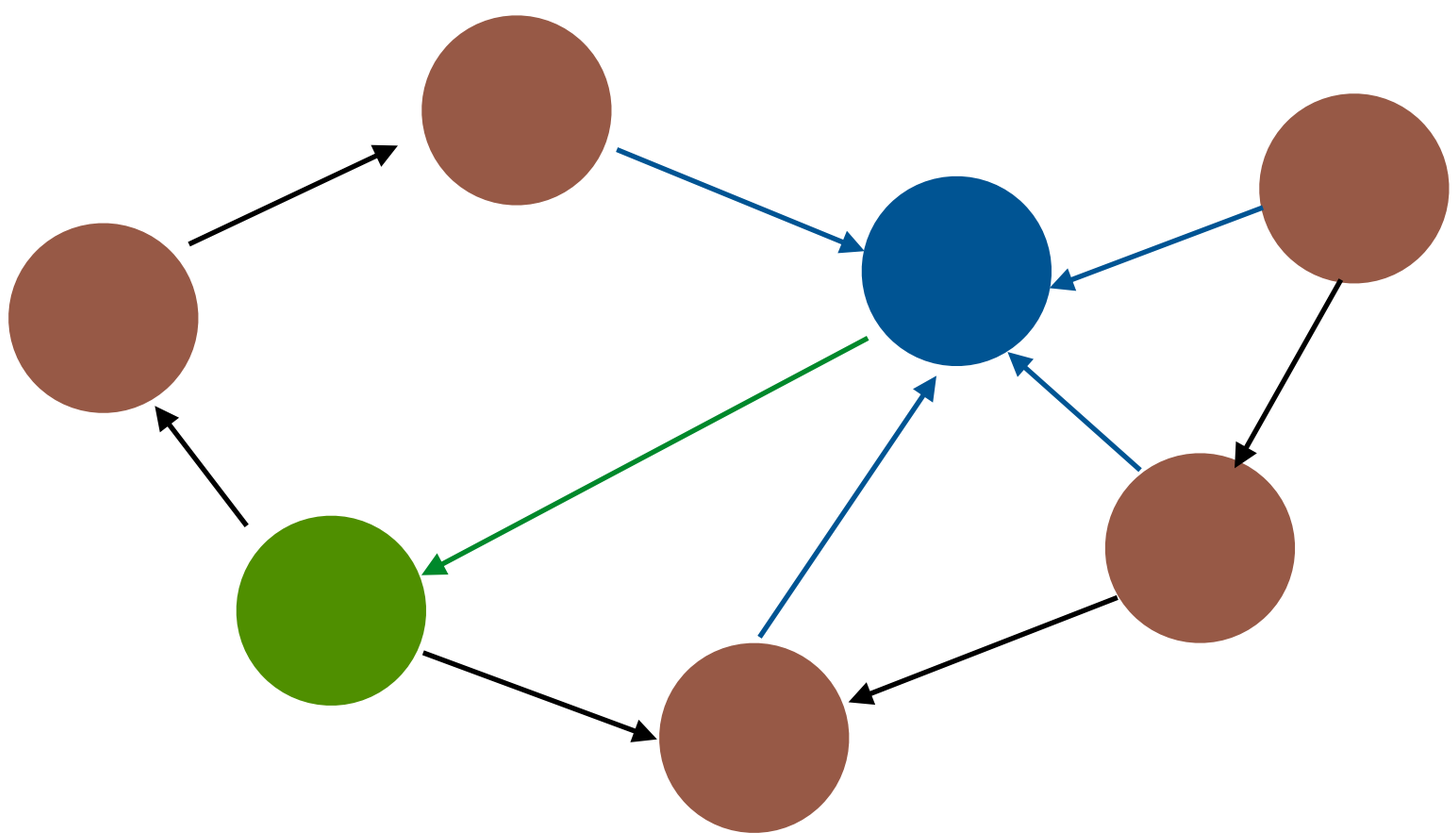
Link analysis for web ranking

- Beyond text content, the authority of a webpage serves as a critical factor for ranking.
  - Evaluates the significance of webpages based on their interlinking, considering the relevance of the source and the target of links.



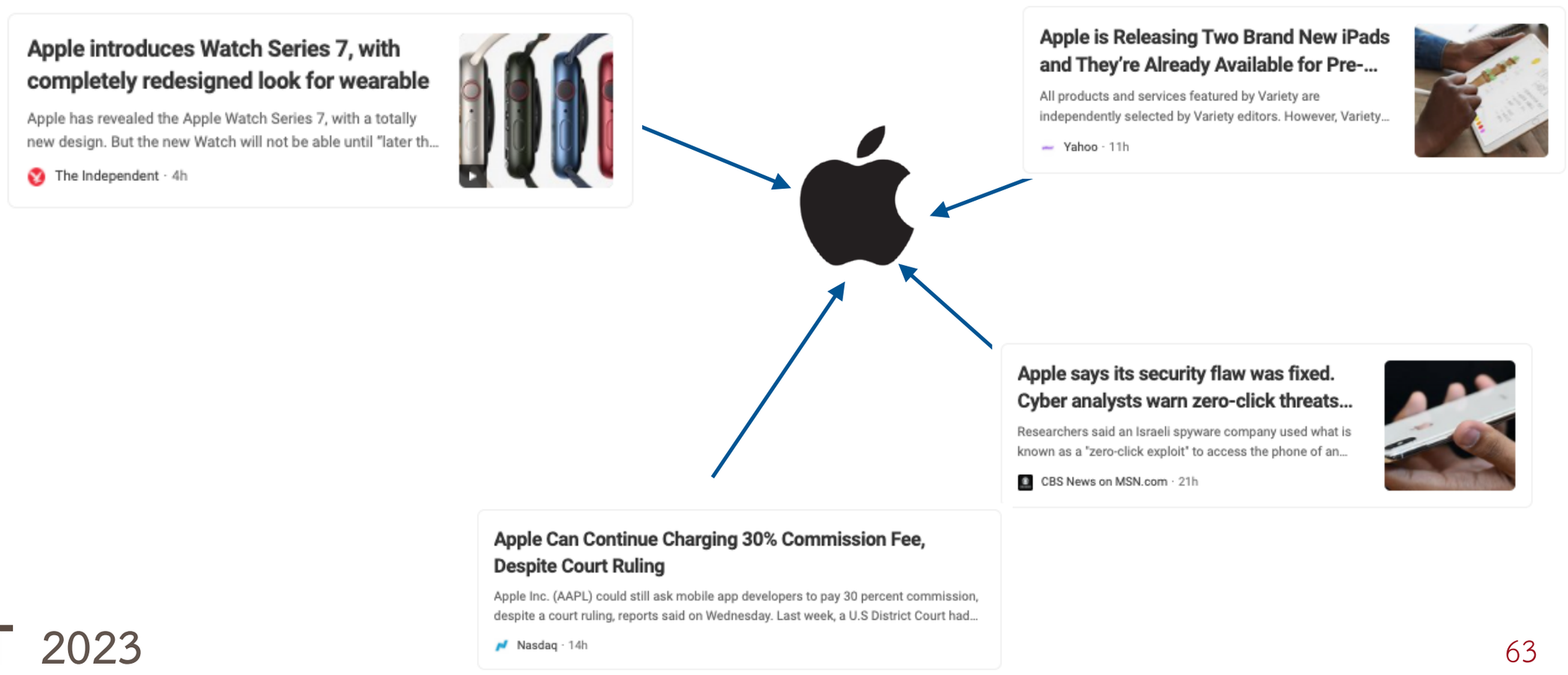
The power of links

- A webpage that many others reference is often seen as a reputable source.
  - Being cited by a prominent source can significantly increase a page's visibility, e.g., if a big name refers to you, you should deserve an attention.

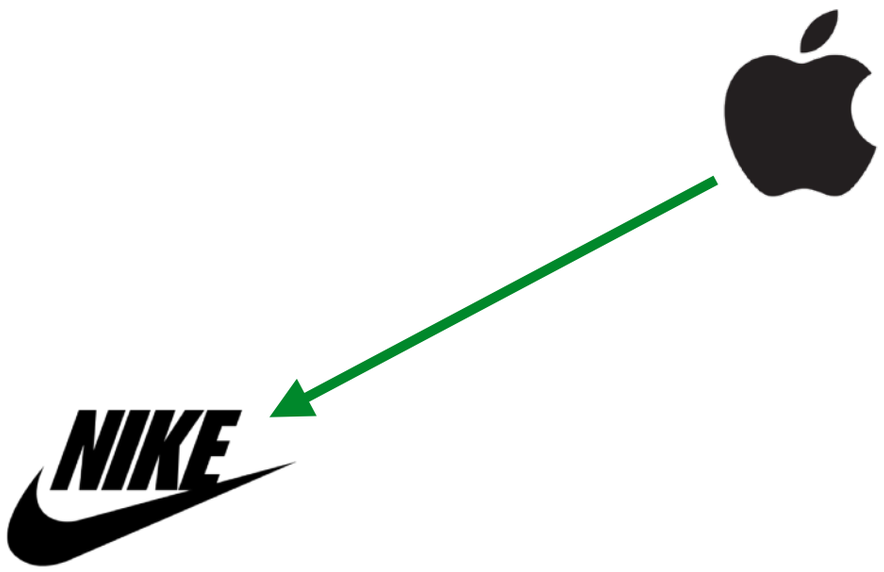


Link analysis

- A **big name** gains its status by the volume of references it receives.
- A mention from a highly-referenced source can confer significant attention to the recipient.
  - The effect is substantial regardless of the referrer's positive or negative standing.



Typically, big names point to big names, e.g.,



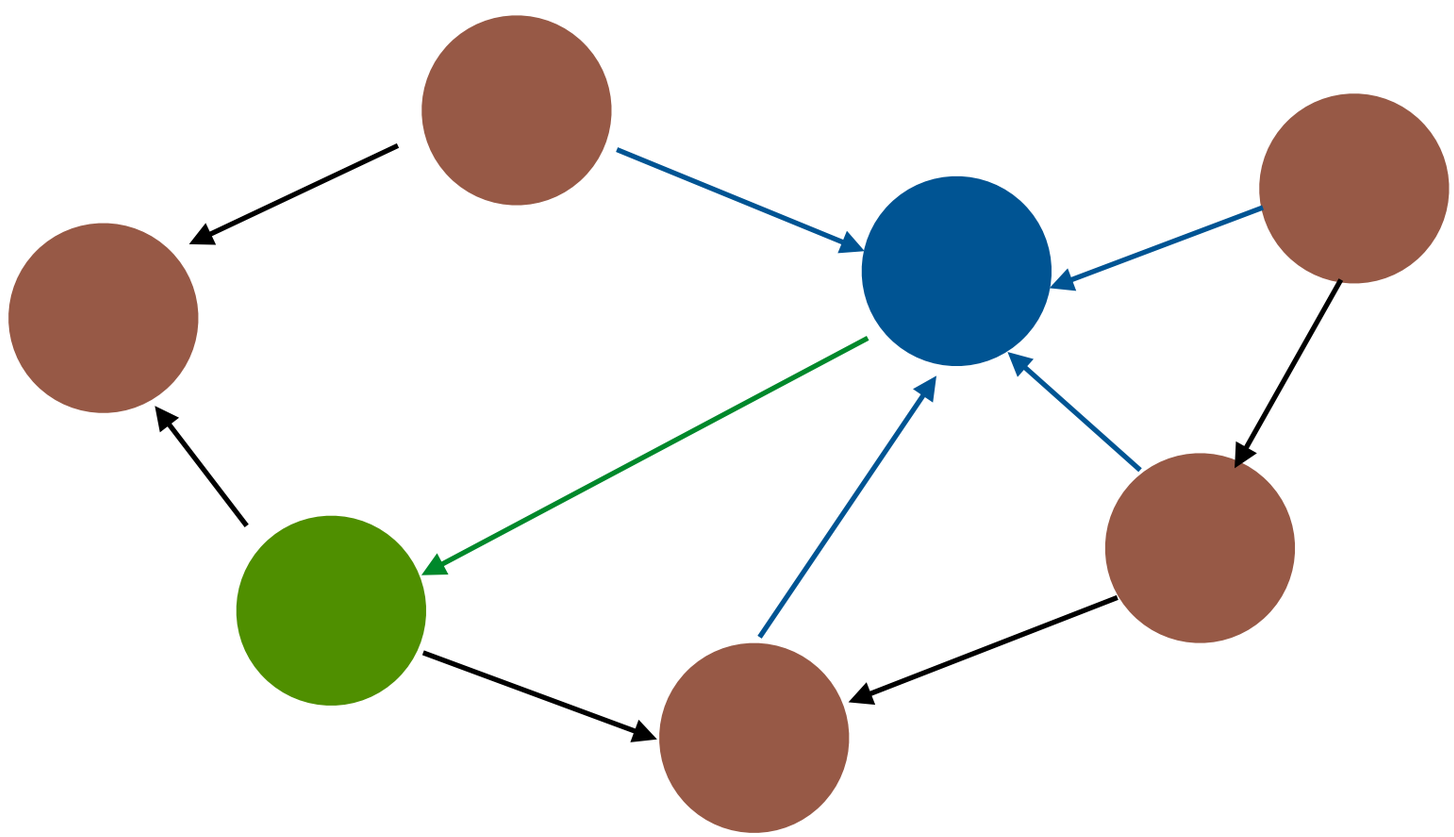
This is typical

But not this


CAMT
College of Arts, Media and Technology
Chiang Mai University

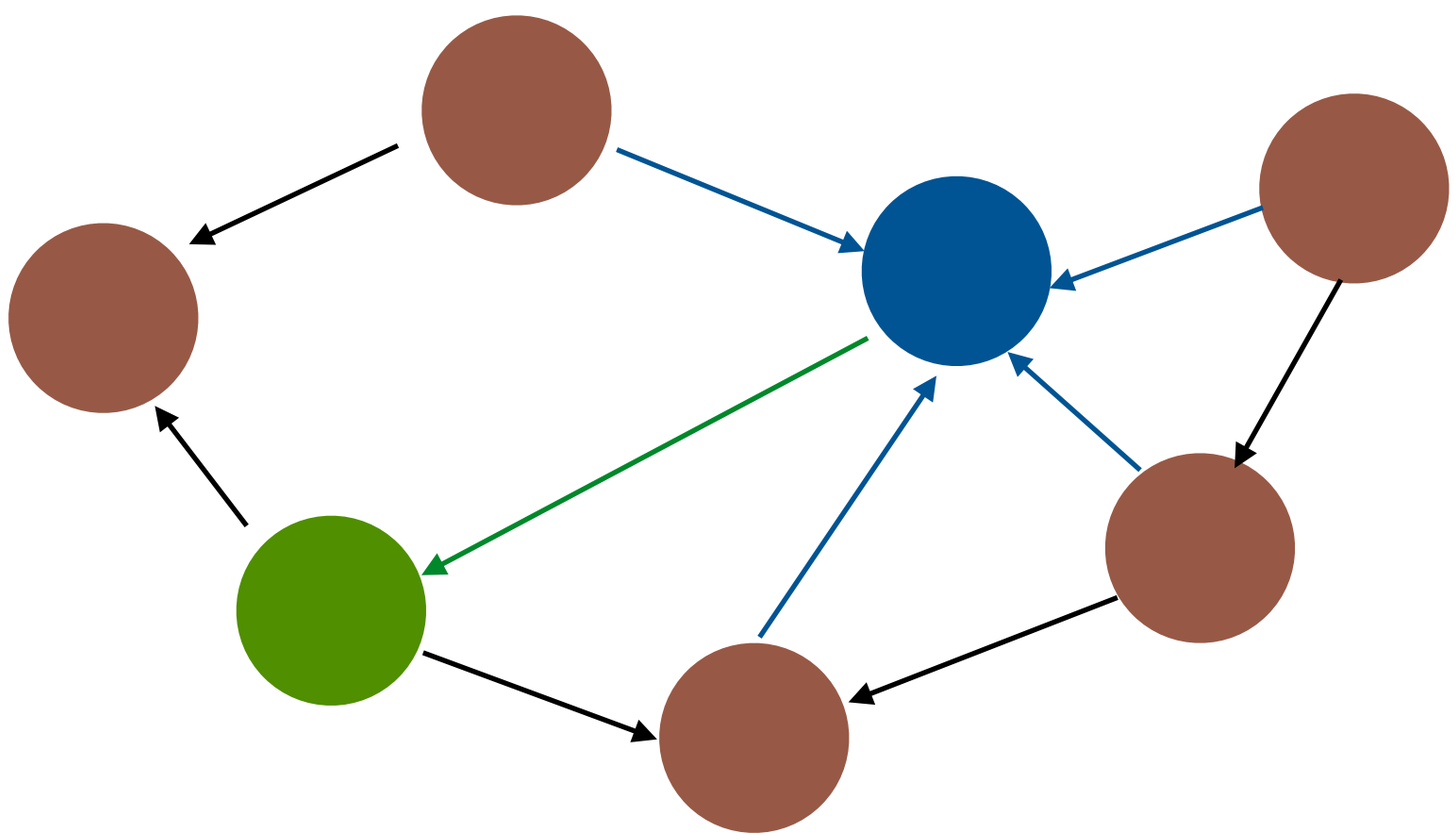
Challenge

- Technically, the true value of a page's score depends on a comprehensive analysis of its **inbound** links.
  - The score a page contributes to another (e.g., from Blue to Green) can only be finalized after evaluating all pages linking to it (blue).



Challenge

- Technically, the true value of a page's score depends on a comprehensive analysis of its **inbound** links.
  - Without a well-defined strategy, the interdependence of page evaluations can lead to a deadlock in scoring.



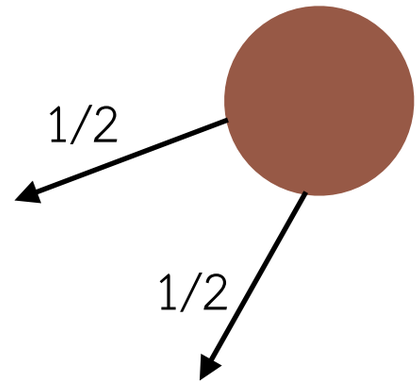
Pagerank scoring

- PageRank addresses the deadlock in link analysis by using a probabilistic model of the web.

- An iterative algorithm distributes the scores of each webpage across the network, refining the scores until they stabilize across the entire web graph.

Pagerank scoring

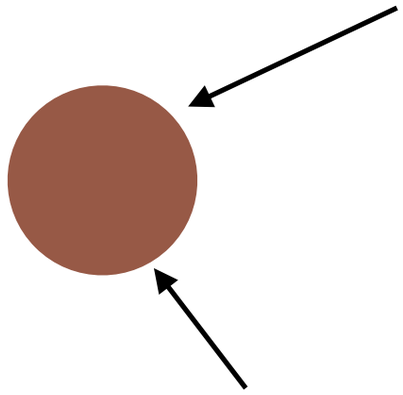
- The original Pagerank assumes that a user has the same chance to click on any link on a page.



- The algorithm predicts that the distribution of probabilities, through iterative propagation, will stabilize over time.

Pagerank scoring

- Web pages often contain dead-ends which halt the flow of PageRank scores, e.g.,

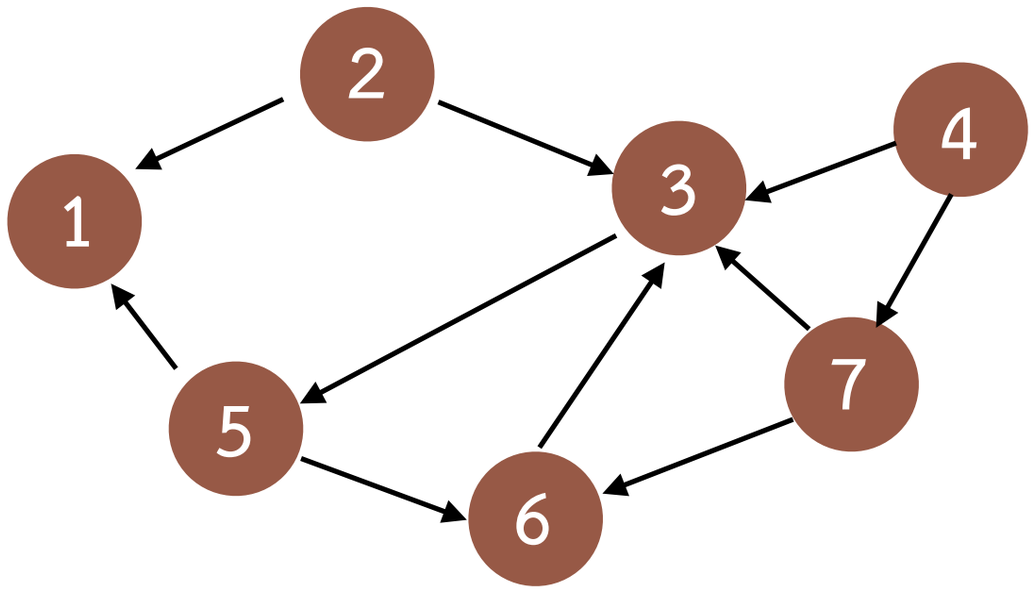


- These dead-ends prevent their scores from being passed on.

Teleporting

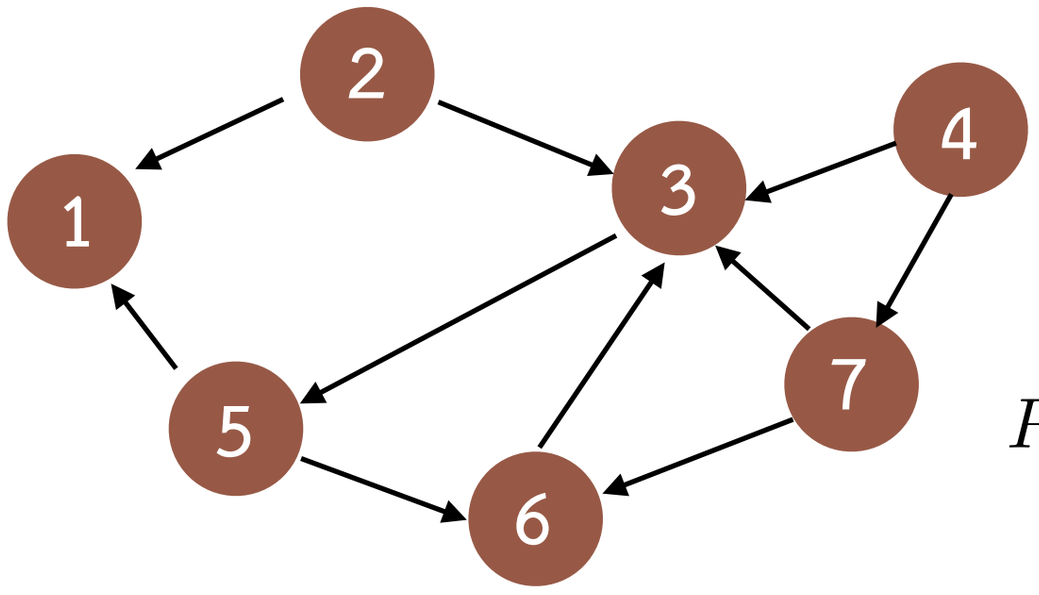
- Teleporting is like imagining users can go to a new page at any time, avoiding stuck pages.
  - This mitigates the issue of dead-ends in the link structure.
- The original PageRank algorithm assigns 85% weight to the link analysis score, with the remaining 15% evenly distributed across all pages, ensuring continuous score propagation.

Example



$$P = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1/2 & 0 & 0 & 0 & 1/2 \\ 1/2 & 0 & 0 & 0 & 0 & 1/2 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/2 & 0 & 0 & 1/2 & 0 \end{pmatrix} \end{array}$$

Fixing teleporting



$$P' = \alpha P + (1-\alpha)ee^T/n, \text{where } \alpha = 0.85$$

$$P' = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\ \begin{pmatrix} 1/7 & 1/7 & 1/7 & 1/7 & 1/7 & 1/7 & 1/7 \\ 25/56 & 3/140 & 25/56 & 3/140 & 3/140 & 3/140 & 3/140 \\ 3/140 & 3/140 & 3/140 & 3/140 & 61/70 & 3/140 & 3/140 \\ 3/140 & 3/140 & 25/56 & 3/140 & 3/140 & 3/140 & 25/56 \\ 25/56 & 3/140 & 3/140 & 3/140 & 3/140 & 25/56 & 3/140 \\ 3/140 & 3/140 & 61/70 & 3/140 & 3/140 & 3/140 & 3/140 \\ 3/140 & 3/140 & 25/56 & 3/140 & 3/140 & 25/56 & 3/140 \end{pmatrix} \end{array}$$

Power iteration

```
01  x0 = np.matrix([1/7] * 7)
02  P = np.matrix([
03                  [1/7, 1/7, 1/7, 1/7, 1/7, 1/7, 1/7],
04                  [25/56, 3/140, 25/56, 3/140, 3/140, 3/140, 3/140],
05                  [3/140, 3/140, 3/140, 3/140, 61/70, 3/140, 3/140],
06                  [3/140, 3/140, 25/56, 3/140, 3/140, 3/140, 25/56],
07                  [25/56, 3/140, 3/140, 3/140, 3/140, 25/56, 3/140],
08                  [3/140, 3/140, 61/70, 3/140, 3/140, 3/140, 3/140],
09                  [3/140, 3/140, 25/56, 3/140, 3/140, 25/56, 3/140],
10                ])
11  print(x0*P)
12  print(x0 * P * P)
13  print(x0 * P * P * P)
```

Power iteration

```
01  x0 = np.matrix([1/7] * 7)
02  P = np.matrix([
03                  [1/7, 1/7, 1/7, 1/7, 1/7, 1/7, 1/7],
04                  [25/56, 3/140, 25/56, 3/140, 3/140, 3/140, 3/140],
05                  [3/140, 3/140, 3/140, 3/140, 61/70, 3/140, 3/140],
06                  [3/140, 3/140, 25/56, 3/140, 3/140, 3/140, 25/56],
07                  [25/56, 3/140, 3/140, 3/140, 3/140, 25/56, 3/140],
08                  [3/140, 3/140, 61/70, 3/140, 3/140, 3/140, 3/140],
09                  [3/140, 3/140, 25/56, 3/140, 3/140, 25/56, 3/140],
10                ])
11
12  prev_Px = x0
13  Px = x0*P
14  i=0
15  while(any(abs(np.asarray(prev_Px).flatten()-np.asarray(Px).flatten()) > 1e-8)):
16      i+=1
17      prev_Px = Px
18      Px = Px * P
19
20  print('Converged in {0} iterations: {1}'.format(i, np.asarray(Px).flatten()))
```

PageRank score for the crawled webpage

```
class Pr:

    def __init__(self, alpha):
        self.crawled_folder = Path(os.path.abspath('')).parent / 'crawled/'
        self.alpha = alpha

    def url_extractor(self):
        url_maps = {}
        all_urls = set([])

        for file in os.listdir(self.crawled_folder):
            if file.endswith(".txt"):
                j = json.load(open(os.path.join(self.crawled_folder, file)))
                all_urls.add(j['url'])
                for s in j['url_lists']:
                    all_urls.add(s)
                url_maps[j['url']] = list(set(j['url_lists']))
        all_urls = list(all_urls)
        return url_maps, all_urls
```

```
if __name__ == '__main__':
    s = Pr(alpha=0.85)
    s.pr_calc()
```

PageRank scores of the crawled webpages

```
01   def pr_calc(self):
02       url_maps, all_urls = self.url_extractor()
03       url_matrix = pd.DataFrame(columns=all_urls,index=all_urls)
04
05       for url in url_maps:
06           if len(url_maps[url]) > 0 and len(all_urls) > 0:
07               url_matrix.loc[url] = (1 - self.alpha) * (1 / len(all_urls))
08               url_matrix.loc[url, url_maps[url]] = url_matrix.loc[url, url_maps[url]] + (self.alpha * (1/
len(url_maps[url])))
09
10       url_matrix.loc[url_matrix.isnull().all(axis=1), :] = (1 / len(all_urls))
11       # print(url_matrix.sum(1).values)
12
13       x0 = np.matrix([1 / len(all_urls)] * len(all_urls))
14       P = np.asmatrix(url_matrix.values)
15
16       prev_Px = x0
17       Px = x0 * P
18       i = 0
19       while (any(abs(np.asarray(prev_Px).flatten() - np.asarray(Px).flatten()) > 1e-8)):
20           i += 1
21           prev_Px = Px
22           Px = Px * P
23
24       print('Converged in {0} iterations: {1}'.format(i, np.around(np.asarray(Px).flatten().astype(float), 5)))
25
26       self.pr_result = pd.DataFrame(Px, columns=url_matrix.index, index=['score']).T.loc[list(url_maps.keys())]
```

PageRank scores of the crawled webpages

```
print(s.pr_result.sort_values(by='score', ascending=False))
```

```
[34]: s.pr_result.sort_values(by='score', ascending=False)
```

[34]:

	score
https://camt.cmu.ac.th/	0.002477
https://camt.cmu.ac.th/index.php/th/	0.002164
https://camt.cmu.ac.th/index.php/en/	0.002138
https://www.cmu.ac.th/th/privacy	0.002131
https://www.google.co.th/maps/@18.8004465,98.950368,18.27z?hl=en	0.002127
...	...
https://cmu.ac.th/cn/faculty/agriculture/aboutus	0.000294
https://cmu.ac.th/th/article/356659b6-9c44-4f15-bf06-b162c17380b1	0.000294
https://cmu.ac.th/cn/content/D7ABB770-4C2B-41EE-825E-720939F6DFD4	0.000294
https://cmu.ac.th/cn/faculty/school_of_public_policy/aboutus	0.000294
https://cmu.ac.th/th/SDGs/11	0.000294

Integrating the PageRank score with Elasticsearch

```
01  class Indexer:
02
03      def __init__(self):
04          self.crawled_folder = Path(__file__).parent / '../crawled/'
05          with open(self.crawled_folder / 'url_list.pickle', 'rb') as f:
06              self.file_mapper = pickle.load(f)
07          self.es_client = Elasticsearch("localhost:9200", basic_auth=("elastic",
"7L4*ufX=xVOj7qa9LDj="), ca_certs="~/http_ca.crt")
08
09      def run_indexer(self):
10          self.pr = Pr(alpha=0.85)
11          self.pr.pr_calc()
12          self.es_client.indices.create(index='simple', ignore=400)
13          self.es_client.indices.delete(index='simple',  ignore=[400, 404])
14
15          for file in os.listdir(self.crawled_folder):
16              if file.endswith(".txt"):
17                  j = json.load(open(os.path.join(self.crawled_folder, file)))
18                  j['id'] = j['url']
19                  j['pagerank'] = self.pr.pr_result.loc[j['id']].score
20                  print(j)
21                  self.es_client.index(index='simple', body=j)
```

Change to yours

Calculate the score and the results are stored at self.pr.pr_result

Adding a filed named ‘pagerank’ to the document

Integrating the PageRank score with Elasticsearch

POST localhost:9200/simple/_search Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
{
    "_source": {
        "exclude": [ "url_lists" ]
    },
    "query": {
        "script_score": {
            "query": {
                "match": { "text": "vision" }
            },
            "script": {
                "source": "_score * doc['pagerank'].value"
            }
        }
    }
}
```

Integrating the PageRank with Flask

```
01  from flask import Flask, request
02  from elasticsearch import Elasticsearch
03  import pandas as pd
04  import time
05
06  app = Flask(__name__)
07  app.es_client = Elasticsearch("http://localhost:9200")
08
09  @app.route('/search', methods=['GET'])
10  def search():
11      start = time.time()
12      response_object = {'status': 'success'}
13      argList = request.args.to_dict(flat=False)
14      query_term=argList['query'][0]
15      results = app.es_client.search(index='simple', source_excludes=['url_lists'], size=100,
16          query={"script_score": {"query": { "match": { "text": query_term } }, "script": {"source":
"_score * doc['pagerank'].value"}}}) ◀..... Here
17      end = time.time()
18      total_hit = results['hits']['total']['value']
19      results_df = pd.DataFrame([[hit["_source"]['title'], hit["_source"]['url'], hit["_source"]
['text'][:100], hit["_score"]] for hit in results['hits']['hits']], columns=['title', 'url', 'text',
'score'])
20
21      response_object['total_hit'] = total_hit
22      response_object['results'] = results_df.to_dict('records')
23      response_object['elapse'] = end - start
24
25      return response_object
26
27  if __name__ == '__main__':
28      app.run(debug=True)
```

Integrating the PageRank with Flask

GET http://localhost:5000/search?query=camt Send

Params • Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	ooo	Bulk Edit
query	camt			

Body Cookies Headers (5) Test Results 200 OK 69 ms 36.3 KB Save Response

Pretty Raw Preview Visualize JSON


```
{
    "elapse": 0.05401611328125,
    "results": [
        {
            "score": 0.0029353334,
            "text": "Home About us Back Vision and Mission Map List
                of CAMT's staff ",
            "title": "Home",
            "url": "https://camt.cmu.ac.th/index.php/en/"
        },
        {
            "score": 0.0027341277,
            "text": "หน้าหลัก รู้จักเรา Back วิสัยทัศน์และพันธกิจ
                แผนที่ รายชื่อบุคลา",
            "title": "หน้าหลัก",
            "url": "https://camt.cmu.ac.th/"
        },
```

Integrating the PageRank with Flask


```
// 20231111154616
// http://localhost:5000/search_es?query=school

{
  "elapse": 0.05148172378540039,
  "results": [
    {
      "score": 0.013082746,
      "text": "Choose your language         ไทย  English (UK)                                    หน้าแรก        รับสมัคร",
      "title": "วิทยาลัยศิลปะ สื่อ และเทคโนโลยี",
      "url": "http://go.camt.cmu.ac.th"
    },
    {
      "score": 0.0033173745,
      "text": "<< คลิกที่นี่ >> ระบบรับสมัคร Gifted School | Pre College",
      "title": "Gift School 2023",
      "url": "https://service.camt.cmu.ac.th/gifted"
    },
    {
      "score": 0.0025128145,
      "text": "MIdS : Multidisciplinary and Interdisciplinary School   Chiang Mai University   Admission 2024  Grad",
      "title": "Graduate School, Chiang Mai University",
      "url": "https://www.grad.cmu.ac.th/"
    },
    {
      "score": 0.001840272,
      "text": "Home  Regulations  Knowledge  Lecturers  Structure  Student Status  Tracking System  ISO  Contact us",
      "title": "CAMT : College of Arts, Media and Technology",
      "url": "https://pandit.camt.cmu.ac.th/home/rule"
    },
    {
      "score": 0.0018068758,
      "text": "MIdS : Multidisciplinary and Interdisciplinary School   Chiang Mai University   Admission 2024  Grad",
      "title": "Graduate School, Chiang Mai University",
      "url": "https://www.grad.cmu.ac.th/index.php?lang=en"
    },
    {
      "score": 0.0015487301,
      "text": "Choose your language         lang-active\"> English (UK)                                    "
```

Time for questions

Assignment

- Complete the Flask application
  - Allow the user to search from Elasticsearch and the manually created index (using different route)
  - Integrate PageRank to the search results from the manually created index.
  - Combine the BM25 scores with PageRank
  - Add an html .. tag to the query term and show only two or three sentences surrounding the query term.
  - Discuss about how this new mix of scores makes finding things better or worse.