MET CS688

WEB ANALYTICS AND MINING

ZLATKO VASILKOSKI

WEB MINING

Data Structures

The basic data structures in computer science.

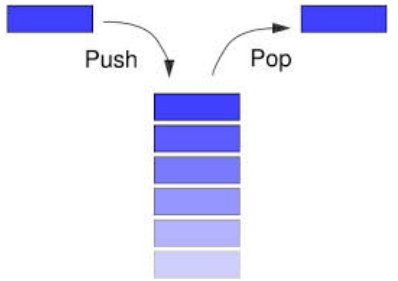
Most of these data structures are used in search, but also in many other different contexts.

- Lists, arrays, hash tables, stacks, queues, graphs, various flavors of trees (binary, balanced) and tries, and heaps.
- Arrays and lists (stacks, queues, hash tables).
- Trees (binary trees, heaps, binary heaps).
  - Binary (each node has at most 2 children) used as a search structure.
  - They can become unbalanced (by insert operations), so that some nodes are deep in the tree, decreasing the search.
  - Balancing a tree ensures that the path length from the root to any leaf node is similar and minimized. Keeping these data structures balanced is very important to keep search cost optimal.
- Tries
- Graphs

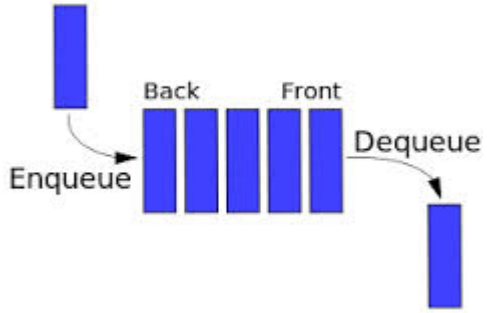
What are Stacks and queues?

Stack is a container of objects that are inserted and removed according to the **last-in first-out** (**LIFO**) principle. For example, stack of books.

- Only two operations are allowed: **push** and **pop**



Queue is a container of objects (a linear collection) that are inserted and removed according to the **first-in first-out** (**FIFO**) principle. For example, a line of students in a cafeteria.

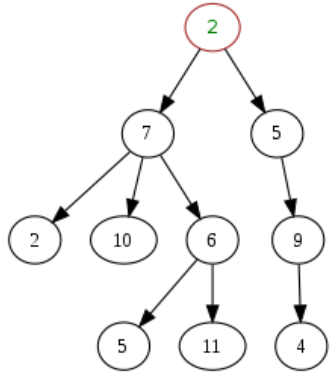


A priority queue is an abstract data type similar to a regular queue or stack data structure in which each element additionally has a "priority" associated with it. In a priority queue, an element with high priority is served before an element with low priority.

What is a Tree, Trie and Graph?

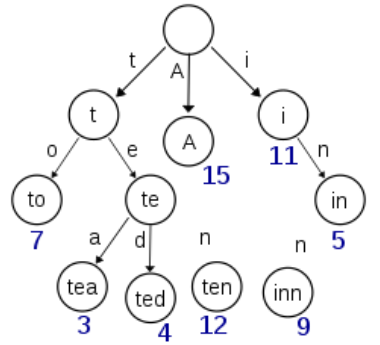
Tree in computer science is a widely used data type for representing a hierarchical tree structure.

- It has a root node and subtrees of children with a parent node, represented as a set of linked nodes.
- The tree data structure can be defined and searched recursively as a collection of nodes (starting from the root node).



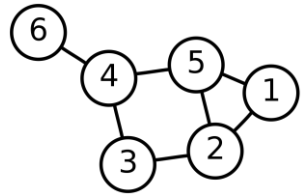
Trie or prefix tree is a type of a tree data structure used in search.

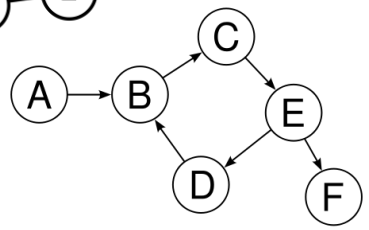
- The keys (the node values) are most often strings.
- The linked nodes (with edges) differ by single character.



Graph is the most general data structure consisting of nodes (vertices) and edges (links between nodes).

- They can be undirected or directed (the edge is an arrow).
- Unlike trees, graphs may contain cycles, so we may come to the same node again. To avoid processing a node more than once, typically a Boolean visited array is used.
- Generally, the word **networks** instead of graphs is used in situations describing transport (sending) things along the links (edges) between the nodes.





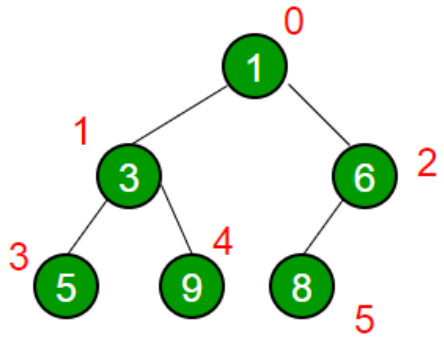
What is a Heap?

A **heap** is an implementation of a data type called a **priority queue**. It is a specialized tree-based data structure (Parent – Child nodes that have values) that satisfies the maximum **heap property** (the minimum is inverse):

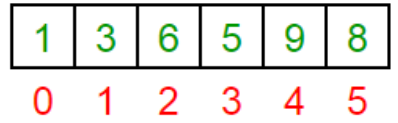
- For any node C, the key (the value) of the parent node P is greater than or equal to the key (the value) of node C.

Heaps are usually implemented with an array, as follows:

- Each element in the array represents a node of the heap, and
- The parent/child relationship is defined implicitly by the elements' indices in the array.
- Level Order is used as a traversal method achieve Array representation (min heap prop)



- Given a node at index i, its children are at indices 2i + 1 and 2i + 2
  - i=2 (node 6), has children at array indices 2i + 1=5 (node 8) and 2i + 2=6 (out)
  - i=1 (node 3), has children at array indices 2i + 1=3 (node 5) and 2i + 2=4 (9)



- This simple indexing scheme makes it efficient to move "up" or "down" the tree.

Heap Sort

Improved variation on **Selection Sort**

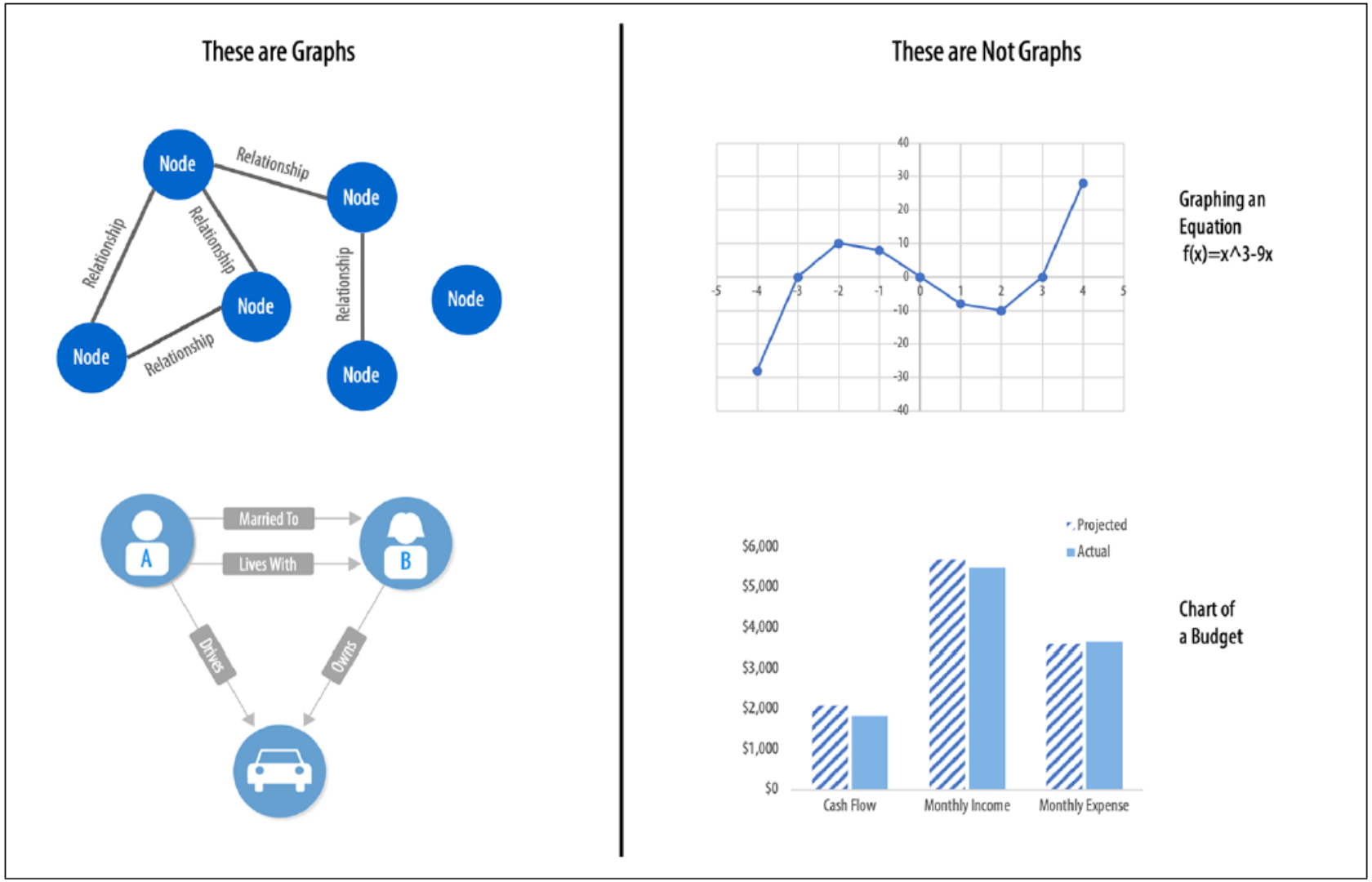
- A sorting algorithm that iterates through the list to ensure every element at index i is the ith smallest/largest element in the list.

6 5 3 1 8 7 2 4

Heap Sort divides its input into a sorted and an unsorted region, and it iteratively shrinks the unsorted region by extracting the largest element from it and inserting it into the sorted region.

- Avoids linear-time scan (Selection Sort).
- Keeps the unsorted region in a heap data structure
  - To find more quickly the largest element in each step.
- Start creating a binary Heap Data Structure from root and whenever larger element is found, replace it upstream the heap (all the way to the root if necessary).

What is a Mathematical Graph?



- Graphs originated (1736) in mathematics.
- They are a way of modeling and analyzing connected data.
- The objects that make up a graph are called nodes or vertices and the links between them are known as relationships, links, or edges.
- One way of looking at graphs is to think of nodes as the nouns in sentences, and edges as verbs giving context to the nodes.

Figure 1-2. A graph is a representation of a network, often illustrated with circles to represent entities which we call nodes, and lines to represent relationships. *Source: "Graph Algorithms" by Mark Needham and Amy E. Hodler*

What is a Graph in Computer Science?

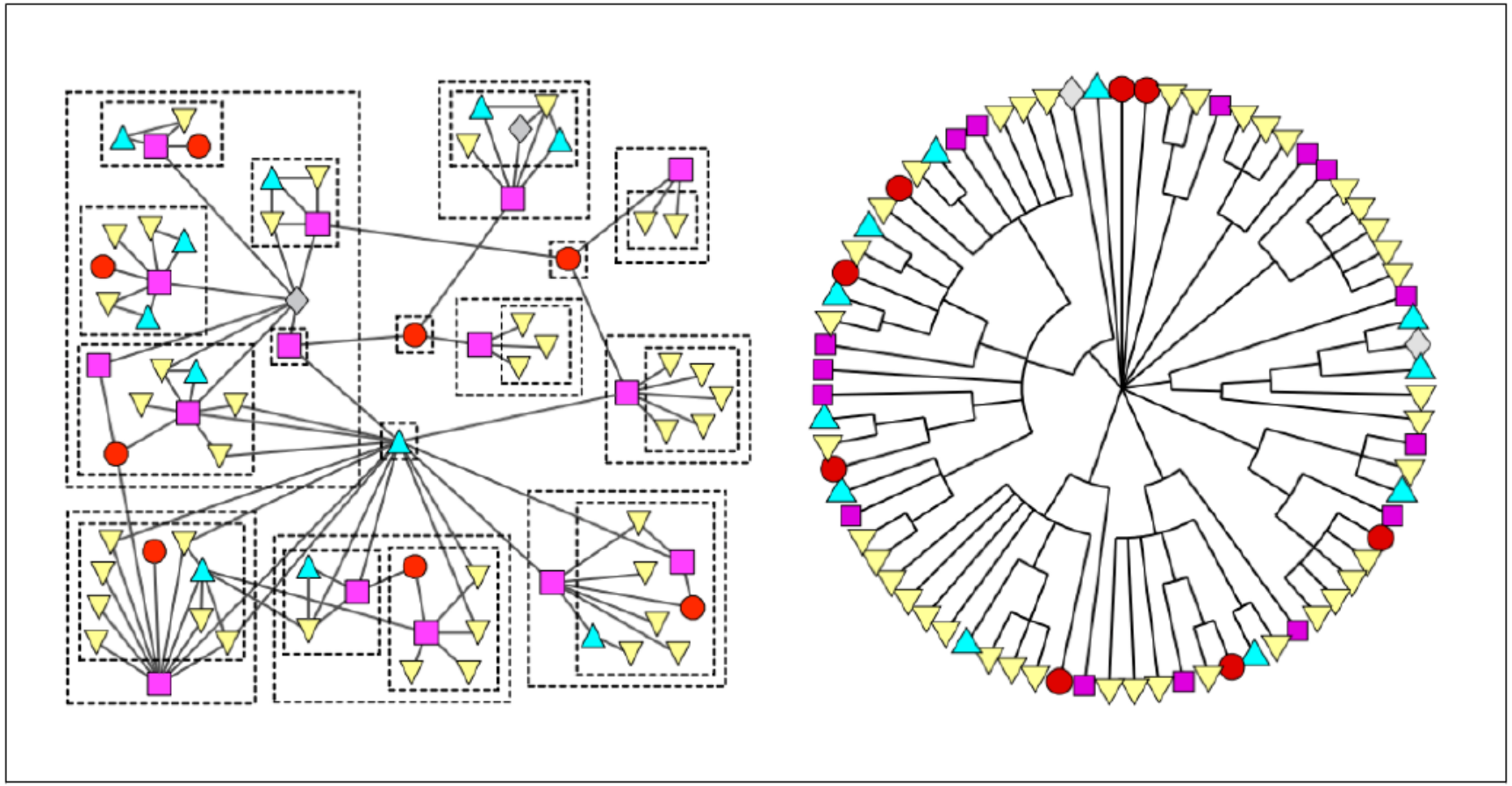


Figure 1-4. This foodweb of grassland species uses graphs to correlate small-scale interactions to larger structure formation.

- Graphs represent the micro and macro scale interactions within global structures and are used to find global patterns in data.
- The graph associations are used
  - In search
  - Databases
  - To forecast behavior and determine missing links.

Source: "Graph Algorithms" by Mark Needham and Amy E. Hodler

Web Mining

- Similar techniques to text mining
  - Difference, in the use of a search engine (the information data is on the web)
- Gathering pages from the web and indexing them in order to support a search engine.
- Web mining technology applies to mining data in a variety of form such as:
  - Web pages
  - A collection of SGML (Standard Generalized Markup Language) generalized markup language for documents
  - XML (Extensible Markup Language) documents, textual data format intended to be both human and machine readable.
  - Genome databases (for example GenBank, PIR)
  - Online dictionary (for example Oxford English Dictionary)
  - Emails or plain texts on a file system.

What is Web Mining?

Discovering information, we need from the World-Wide Web.

- Textual information and web links structure
- Data generated per day is comparable to largest conventional data warehouses
- Often need to react to evolving usage patterns in real-time

Size of the Web

Number of pages – ”infinitely” large.

- Although much of it is duplication (30-40%)
- Best estimate of “unique” static HTML pages comes from search engine claims.
- Google and Yahoo claimed tens of billion (10^9).
- Google recently announced that their index contains 1 trillion (10^{12}) pages.

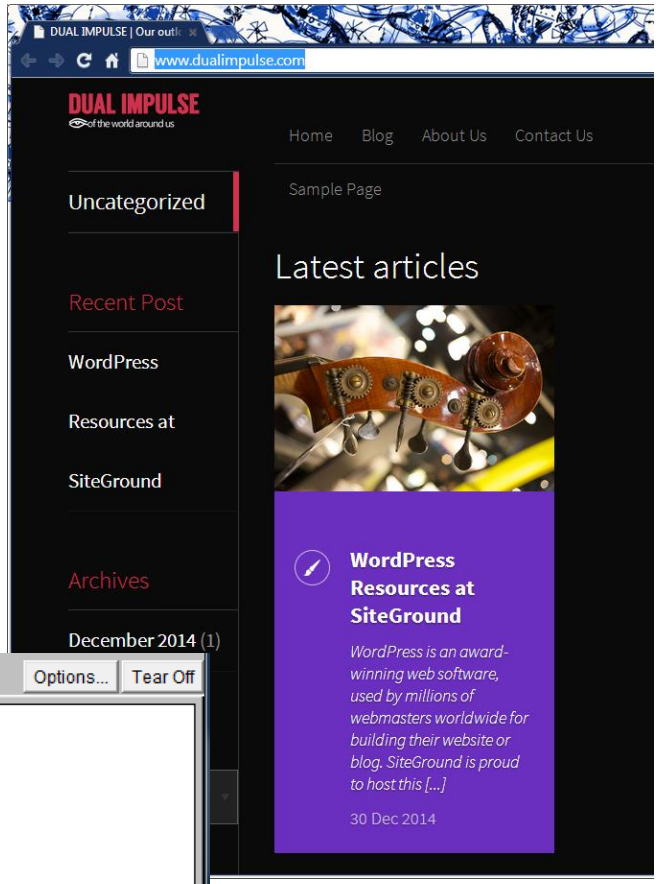
The web as a graph

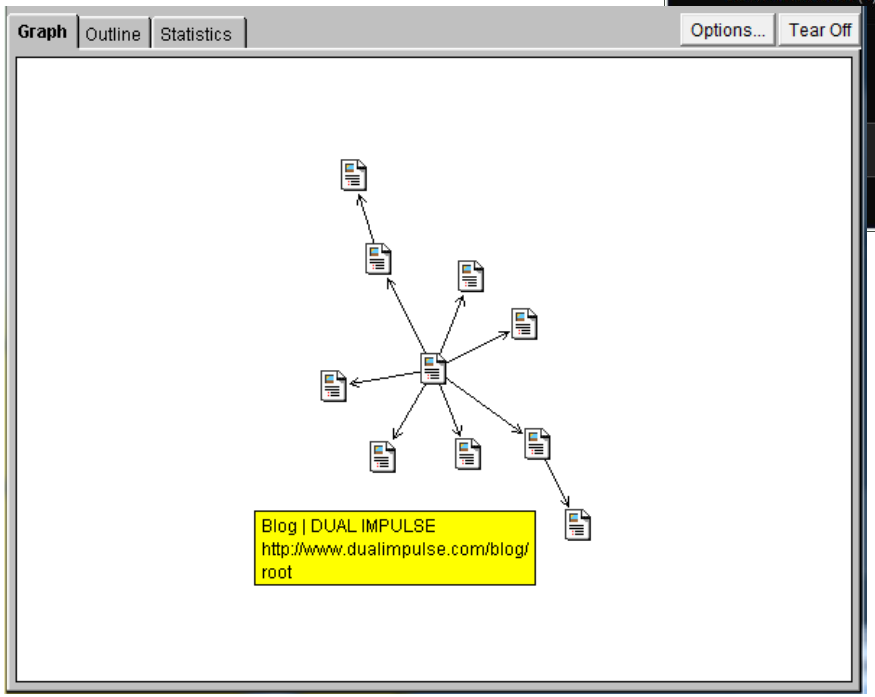
- The Web is a directed graph
  - Pages - nodes,
  - hyperlinks - edges

- Assigning properties to the edges of a graph, such as a web page endorsements turns the graph into a **network**.

- The empirical study of networks has played an important role from biological and social to telecommunication, computer and web networks.

- The web has high linkage 10-20 links/page on average
  - The links between the web pages are not randomly distributed
  - Power-law degree distribution.





Power-law distribution

- A **power law** (also called a **scaling law**) describes the relationship between two quantities x and y, where y varies as some power s of x ($y = x^s$).
  - For example, the volume V of a cube is related to the length a of its sides by a power of 3.

$$V = a^3$$

- There are various power laws in the natural world and networks.
  - A well-known example is the Pareto distribution or "80/20 rule," originally used to describe the situation where 20% of a population controlled 80% of the wealth.
  - Zipf's Law $f_k = 1/k^s$

- In the case of the Web, the links between the web pages are <u>not randomly distributed</u>.

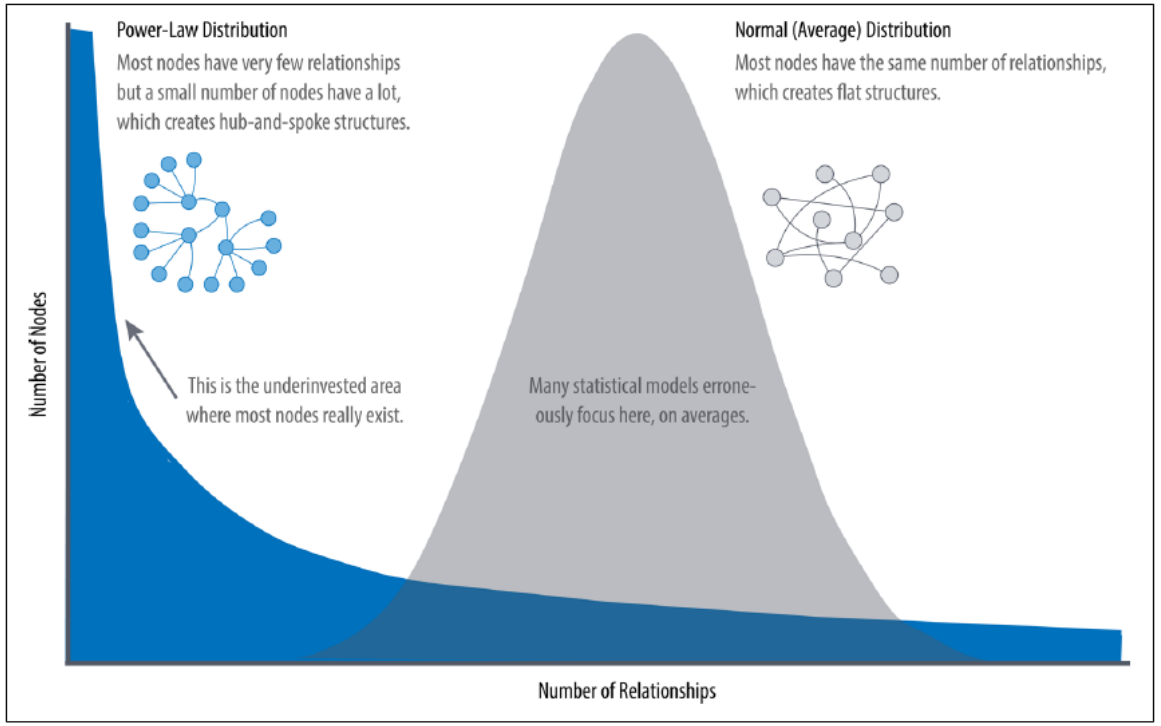


Figure 1-8. Real-world networks have uneven distributions of nodes and relationships represented in the extreme by a power-law distribution. An average distribution assumes most nodes have the same number of relationships and results in a random network.

"**There is no network in nature that we know of that would be described by the random network model.**"

Albert-László Barabási, Director, Center for Complex Network Research, Northeastern University, and author of numerous network science books

Source: "Graph Algorithms" by Mark Needham and Amy E. Hodler

Degree Centrality

- **Degree Centrality** counts the number of incoming and outgoing relationships from a node and is used to find popular nodes in a graph. Degree Centrality was proposed by Linton C. Freeman in his 1979 paper "Centrality in Social Networks: Conceptual Clarification".
- Connectedness to other nodes in the graph is measured with the degree of a node, which is the number of direct relationships a node has with other nodes.
  - Related to "popularity" of individual nodes
  - A person with a high degree in an active social network would have a lot of immediate contacts.
  - It is calculated for **in-degree** and **out-degree**.
- The average degree of a network is simply the total number of relationships divided by the total number of nodes.
- The **degree distribution** is the probability that a randomly selected node will have a certain number of relationships.

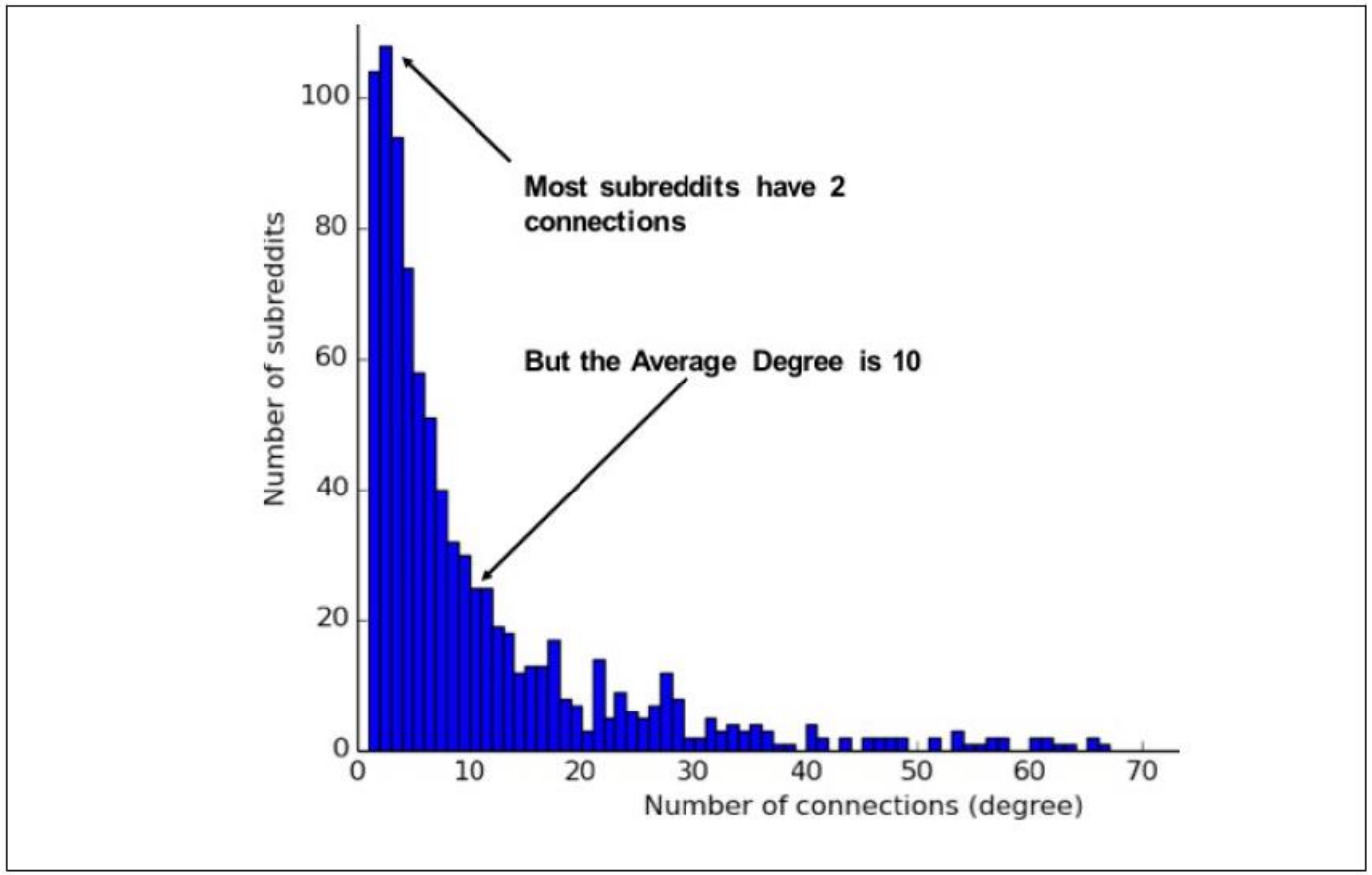


Figure 5-3. This mapping of subreddit degree distribution by Jacob Silterrapa provides an example of how the average does not often reflect the actual distribution in networks. CC BY-SA 3.0.

Source: "Graph Algorithms" by Mark Needham and Amy E. Hodler

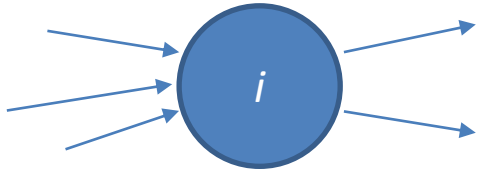
Degree Centrality

- Degree Centrality is used to analyze influence by looking at the number of incoming and outgoing relationships to:
  - Find the “popularity” of individual nodes.
  - Also applied to global analysis when you want to evaluate the minimum degree, maximum degree, mean degree, and standard deviation across the entire graph.

- Example use cases:
  - Identifying powerful individuals though their relationships, such as connections of people in a social network.
  - Separating fraudsters from legitimate users of an online auction site. The weighted centrality of fraudsters tends to be significantly higher due to collusion aimed at artificially increasing prices.

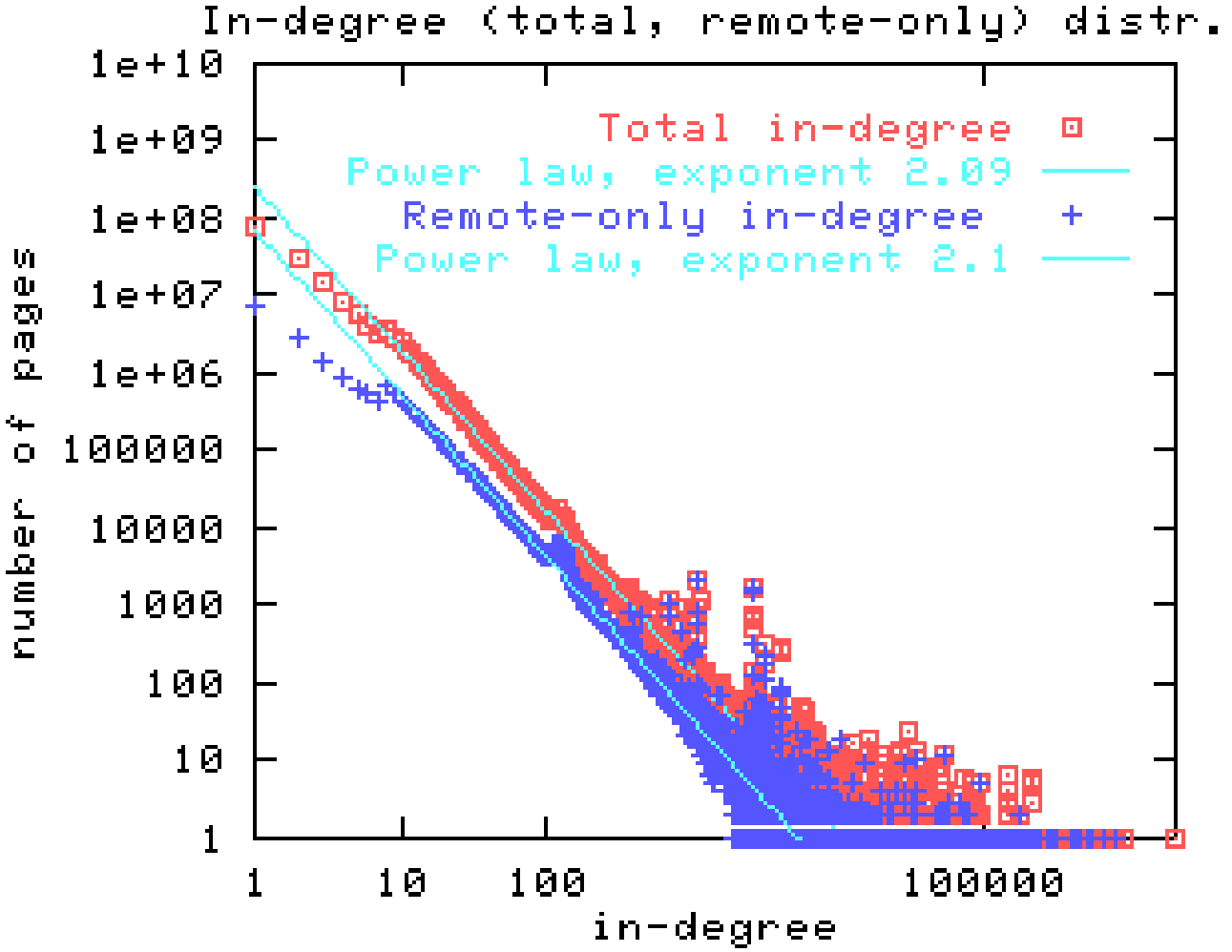
Power-law degree distribution

- They follow a power law distribution based on the **in-node degree** (i).



- The total number N of web pages with in-degree i, almost as a natural law, is proportional to $N \propto i^{-\beta}$, where $\beta \approx 2.1$, strongly indicating some aspects about the association structure of the Web which is spontaneous.

- The log-log plot on the right expresses the power law relationship between the total number of web pages vs. in-degree i



Source: Broder et al, 2000

Approaches to Analyzing Data

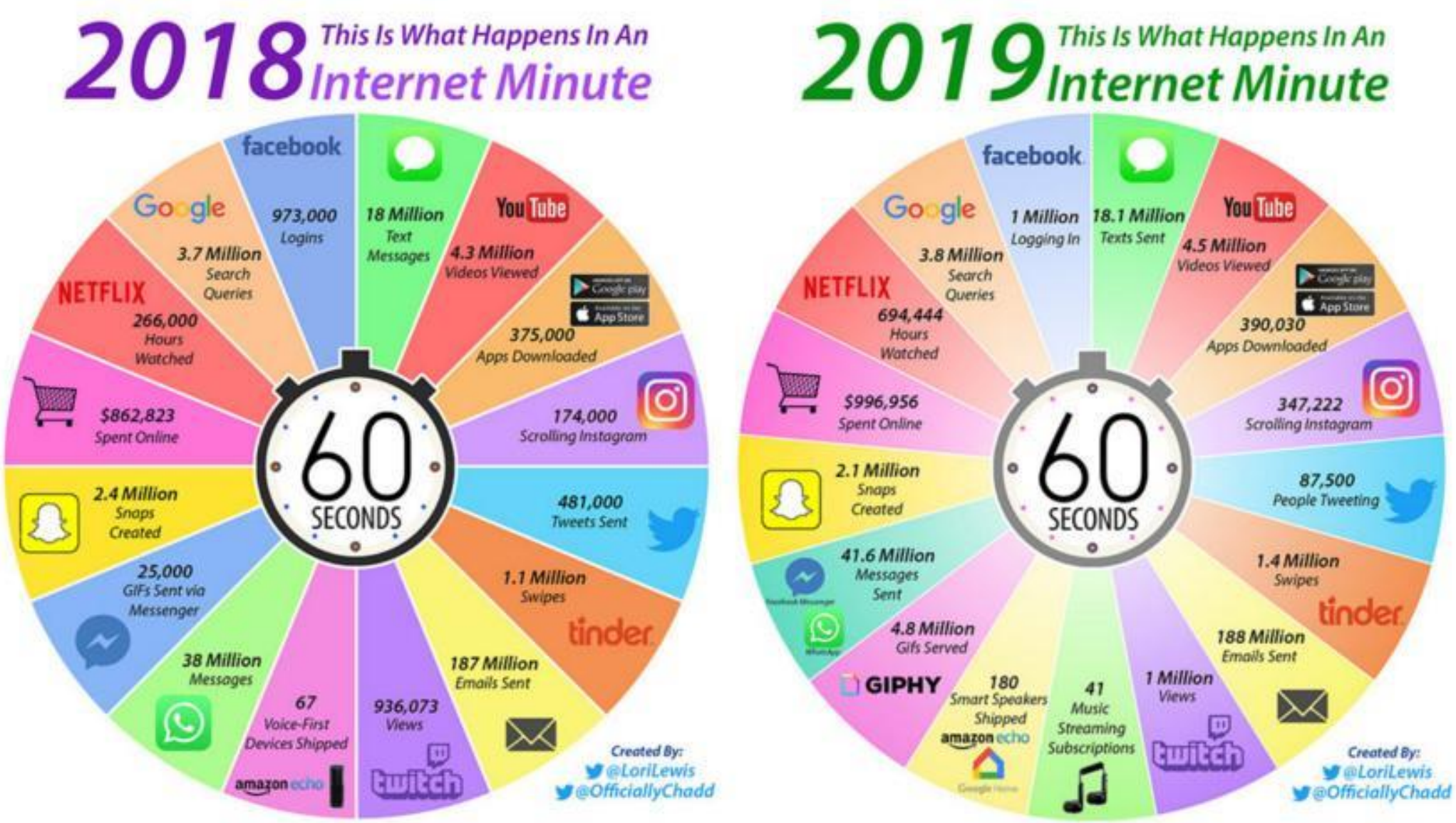
What kind of algorithms we use depends on the kind and the amount of data.

1. Machine Learning approach
   - Using sophisticated algorithms that require relatively long processing time
   - Data sets need to be relatively small, to avoid memory and speed issues.

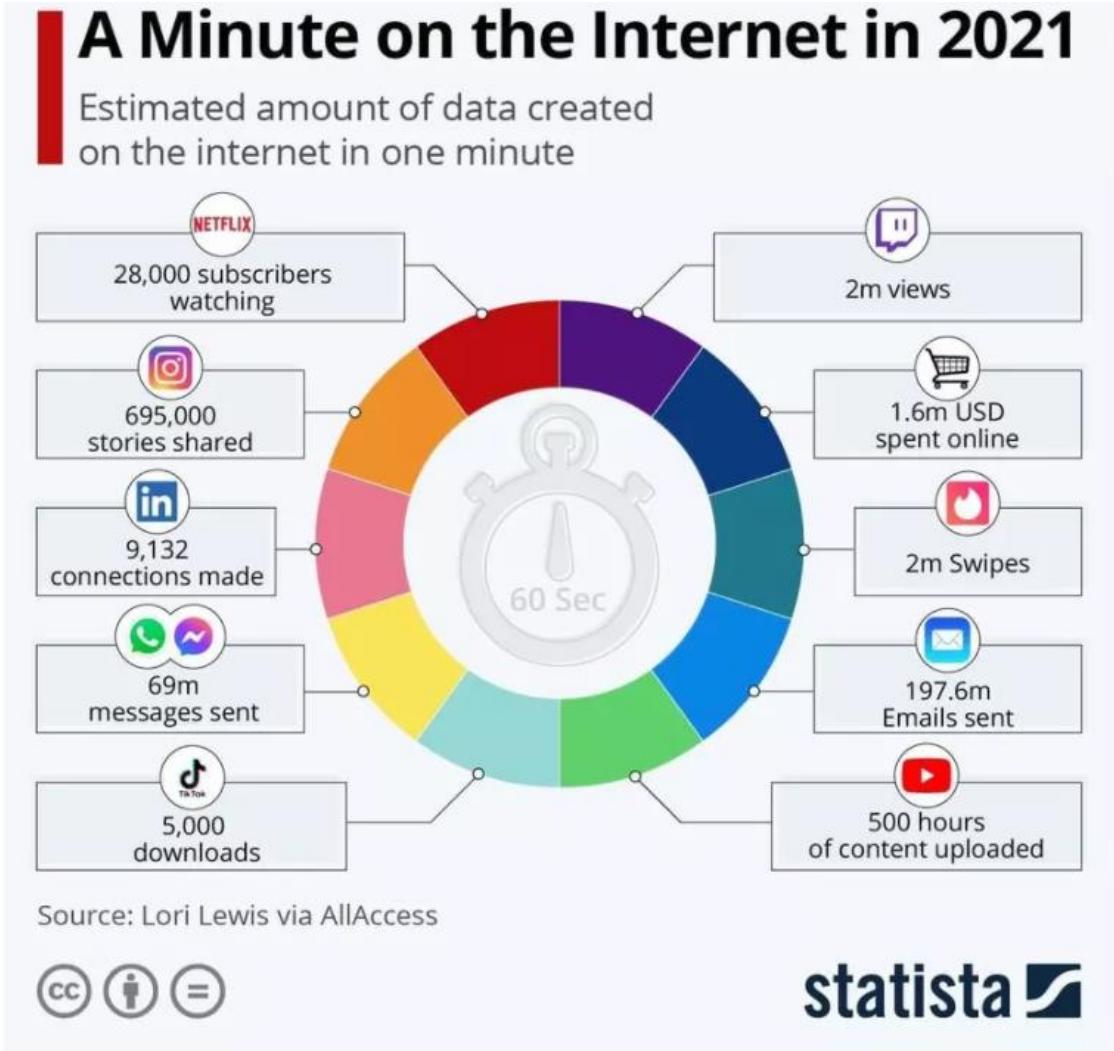
2. Data Mining approach
   - Convenient for big data sets
   - Data size and processing speed forces the use of simpler algorithms.

- In many cases, adding more data leads to better results than improving algorithms but up to some point.

What is happening in an Internet Minute



What is happening in an Internet Minute



What is “Singles Day”?

Singles Day sales frenzy surpasses records (https://www.bbc.com/news/business-46168996)

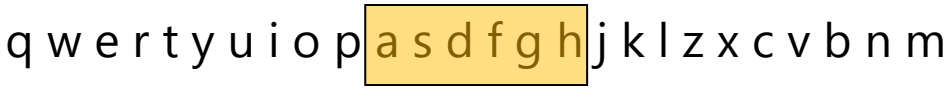
- Alibaba invented the occasion in 2009 to celebrate the unattached as an antithesis to the romantically involved on Valentine's Day.

- According to Bloomberg, it is now the world's biggest online sales event. The 2019 sales total was larger than the combined totals on Black Friday and Cyber Monday's.
  - It had 120,000 orders/minute.
  - ¾ of purchases were made by phone.

- The shopping frenzy has broken world records in e-commerce sales - surpassing previous record at 17:34 Hong Kong Time (10:34 GMT on November 11, 2018).

Big Data Example - Sliding Windows

Model of stream processing



- Queries refer to a window of length N (process the N=6 most recent elements received)

Big Data model extension (lots of online retailer examples)

- N is typically so large that the data cannot be stored in memory, or even on a disk.
- There are so many streams that windows for all cannot be stored

To illustrate the complexity, let's do the following "Alphabet" simplification

- Consider the window of length N to represent the number of sales.
- Use binary stream (1 - for product sold in the n-th transaction)
- Query - how many sold items in a window of sales.

Big Data Example - Sliding Windows

0 1 0 0 1 1 0 1 1 1 0 1 0 1 0 1 1 0 1 1 0 1 1 0

- For a small window of length N=6 we can store the most recent N bits.
- For a "big data" window we cannot do this, for example N=1billion.

Note: We **cannot** get an exact answer without storing the entire window!

- But if approximate answer is acceptable, we can use several simple algorithms.
  - For example, keep moving average of "0" and "1". Important things to consider:
    - How many 1's are in the last N bits?
    - **Question**: What are the assumptions of this simple algorithm on the streaming of "0" and "1" in this approach?
      - Similar number of "0" and "1" in the stream!

Big Data Example - Real-Time Bidding

Real-Time bidding (**RTB**)

- Similar to financial (stock) markets,
- Refers to the online auction process of buying and selling of online ads done in real-time (during those few ms while the page is loading).
  - These auctions are held while the visitor's web page is loading and the RTB algorithm decides which ad, based on visitor's browser history, suits the visitor best, along with the price for displaying the ad.

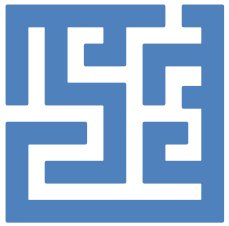
Big Data Example - Real-Time Bidding

Real-Time bidding (**RTB**) facts:

- It was a $24 billion industry in 2018 in US only.
- Advertising buyers bid on visitor to a publisher's web site before the visitor opens the page.
- If the bid is won, the buyer's ad is instantly (within ms) displayed on the publisher's site.
- RTB lets advertisers manage and optimize ads from multiple ad-networks, allowing them to create and launch advertising campaigns, prioritize networks, and allocate percentages of unsold inventory, known as **backfill**.
- There are challenges with mobile implementation since it is more difficult to determine a visitor's browser history on mobile devices.

Question and Answering Systems (Q & A)

Question/Answering engages customer through conversation and makes **Information Retrieval** (IR) experience better



Takes complex questions and meets them with direct, specific answers in the form of a rich snippet.

Helps customer find information faster and saves time to click through and scroll a long document.

Q&A Summary - The next big thing

- Why use Question and Answering (Q&A) systems:
  - To enhance customer's Information Retrieval (IR) experience
  - Provide specific answers to complex questions
  - find most relevant information from very large harmonized corpus

- Integrating ML with Q&A systems require state-of-the-art general-purpose architectures for Natural Language Understanding (NLU).

- Recently developed Q&A ML models are capable to learn basic language modeling by being pre-trained on massive unsupervised training sets.
  - These ML models support over 100 languages

- In order to perform well at specific tasks (like Q&A), they must be trained further -- **fine-tuned** -- on custom data (with significant effort - 1.75 hours per epoch).

- But the repository for pre-trained and fine-tuned models contributed from the wide NLP community is growing every day.

Question and Answering System - Example

Question and Answering System - From a body of text, provide a reasonable answer to a question related to it.

This is a screenshot of the Python code from one such example text on the ancient kingdom of Macedonia.

- On the question:
  - “Who Ruled Macedonia?”
- It identifies exactly what the answer is
  - “the Argead dynasty”

How does it work?

1. Tokenize the input and turn it into tensors
   - Both the question and the text of the content
2. Obtain model scores for each token
3. Get the answer span
   - The model returns 'answer start' and ‘end ' scores for each word in the text
   - Then all the tokens between them are taken and converted back from tensors to words

```
question = "Who ruled Macedonia"

context = """Macedonia was an ancient kingdom on the periphery of Archaic and C
and later the dominant state of Hellenistic Greece. The kingdom was founded and
by the Argead dynasty, followed by the Antipatrid and Antigonid dynasties. Home
Macedonians, it originated on the northeastern part of the Greek peninsula. Bef
century BC, it was a small kingdom outside of the area dominated by the city-st
Sparta and Thebes, and briefly subordinate to Achaemenid Persia."""


# 1. TOKENIZE THE INPUT
# note: if you don't include return_tensors='pt' you'll get a list of lists whi
# exploration but you cannot feed that into a model.
inputs = tokenizer.encode_plus(question, context, return_tensors="pt")

# 2. OBTAIN MODEL SCORES
# the AutoModelForQuestionAnswering class includes a span predictor on top of t
# the model returns answer start and end scores for each word in the text
answer_start_scores, answer_end_scores = model(**inputs)
answer_start = torch.argmax(answer_start_scores)  # get the most likely beginni
answer_end = torch.argmax(answer_end_scores) + 1  # get the most likely end of

# 3. GET THE ANSWER SPAN
# once we have the most likely start and end tokens, we grab all the tokens bet
# and convert tokens back to words!
tokenizer.convert_tokens_to_string(tokenizer.convert_ids_to_tokens(inputs["inpu
```

```
'the Argead dynasty'
```

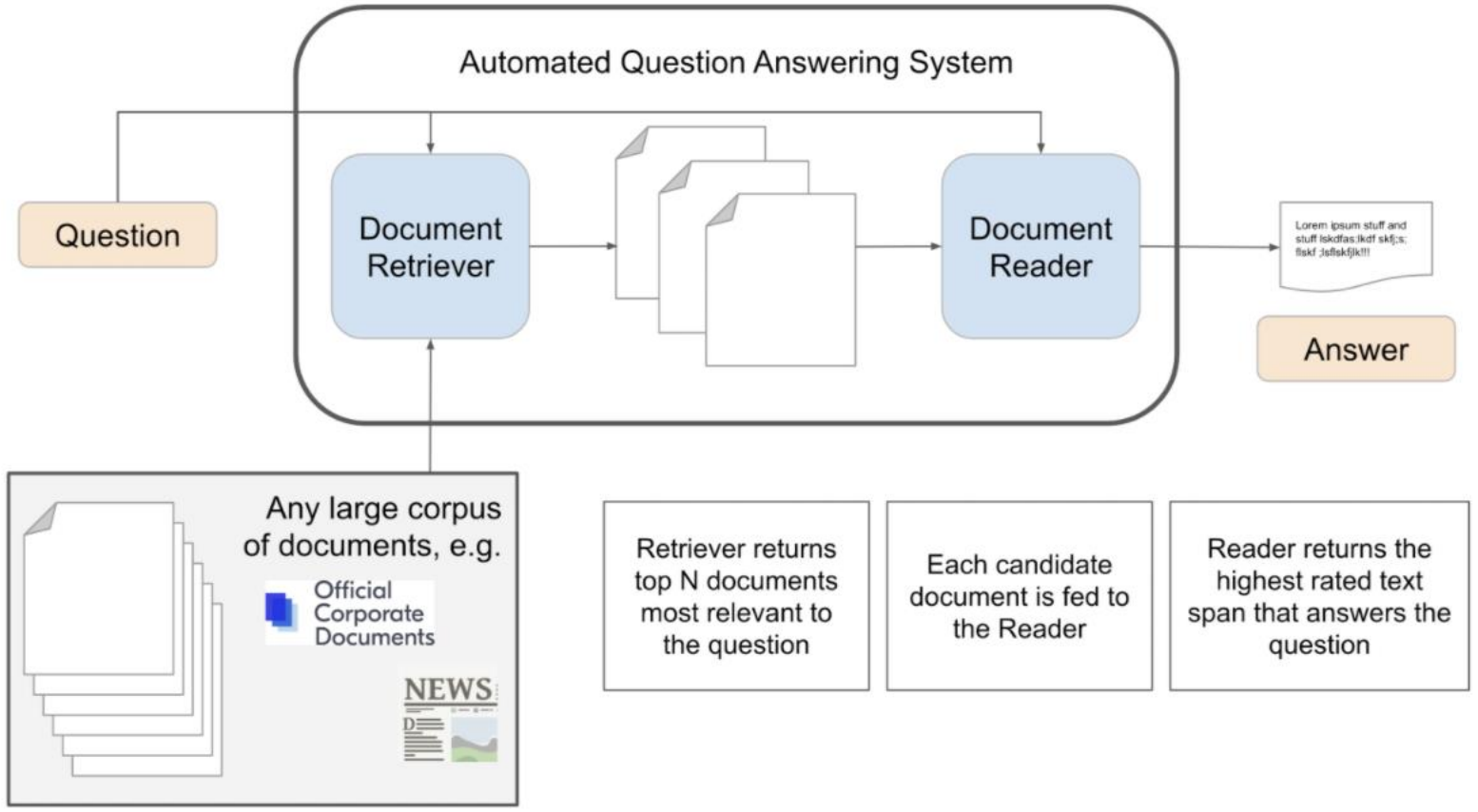
Building a Question and Answering System

Typically, these systems require two main components:

1. ***Document Retriever*** (a search engine) that filters the n most relevant documents from a large collection.
2. ***Document Reader*** that implements ML to these candidate documents to extract the explicit answers.

The Hugging Face (HF) Transformers

- Provide state-of-the-art general-purpose architectures for NLU and NLG.
- All the standard models that HF supports (in over 100 languages) have been pre-trained, which means they've all been fed massive unsupervised training sets in order to learn basic language modeling.
- In order to perform well at specific tasks (like Q&A), they must be trained further -- fine-tuned -- on custom data (with significant effort - 1.75 hours per epoch).
- HF hosts a repository for pre-trained and fine-tuned models contributed from the wide NLP community.



IR-based automated question answering workflow

Automatic Q & A System in a Nutshell

Question => 1) Doc. Retriever => 2) Doc. Reader => Answer

- 1) Doc. Retriever – Return top N most relevant **candidate documents** from the entire corpus
- Each **candidate document** one by one is fed to the Document Reader for processing
- 2) Doc. Reader – Processes single document and returns the most relevant text from that single document related to the question.

1) Doc. Retriever – Various tools to rank matching documents according to their relevance
   - **Challenge: How to select few relevant documents from very large content in reasonable time.**
   - Traditionally the following types of algorithms are used:
     - TF-IDF
     - BM25 (Best Matching) algorithm.
     - Dense Retriever algorithm is a new algorithm that outperforms BM25 by using embeddings which are learned from a small number of questions.

2) Doc. Reader – Uses BERT ML models to semantically provide the answer from each document.
   - **Challenge: Works slow but very well. There are ways to speed it up.**

Illustration

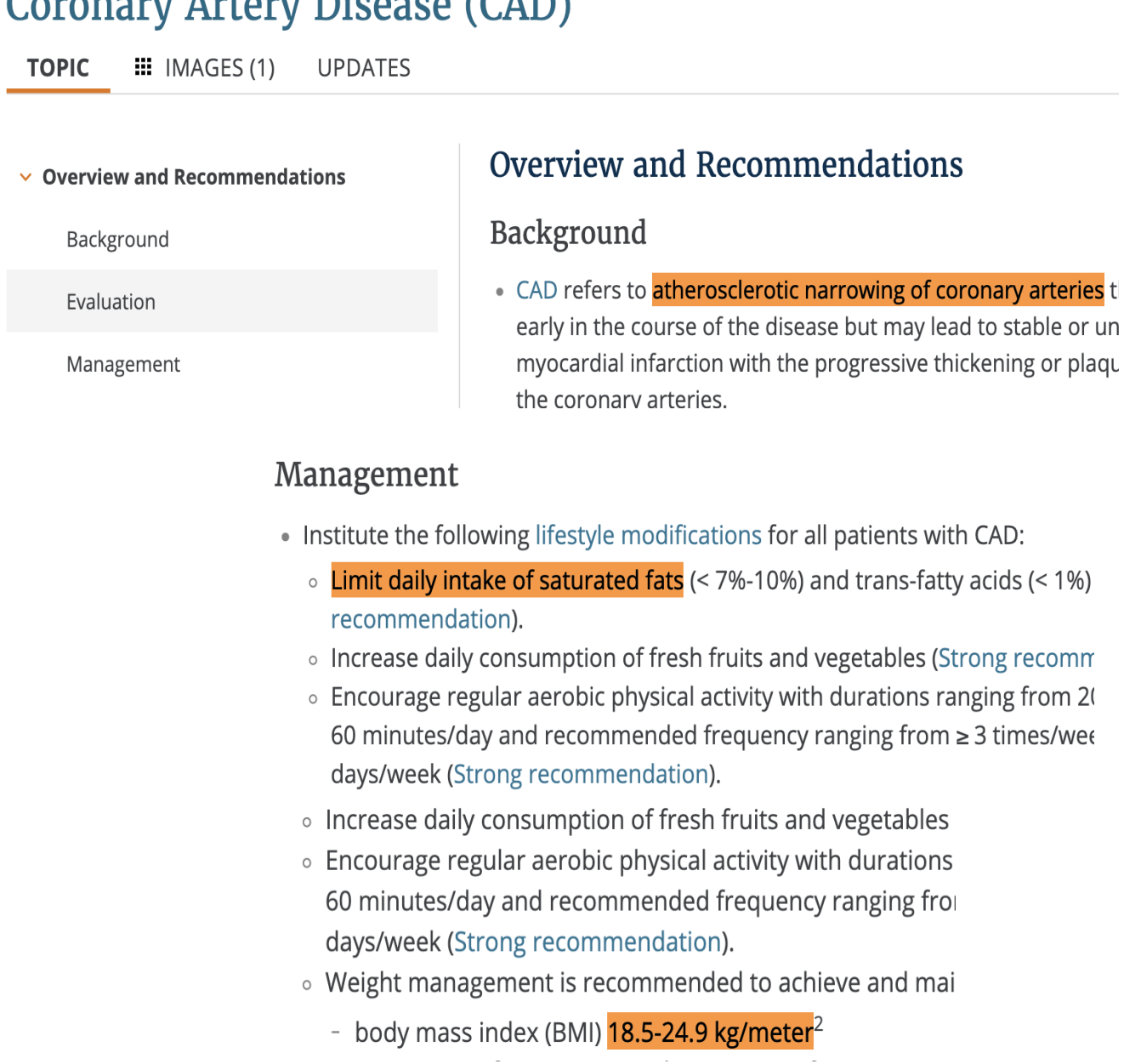
Several answers to questions related to the DynaMed (https://www.dynamed.com/) topic *Coronary Artery Disease (CAD)*

Question: What does CAD refer to?
Answer: atherosclerotic narrowing of coronary arteries

Question: What are the lifestyle modifications for the patients with CAD?
Answer: Limit daily intake of saturated fats (< 7 % - 10 %) and trans - fatty acids (< 1 %)

Question: What are the weight management recommendations for the body mass index (BMI) in patients with CAD?
Answer: 18. 5 - 24. 9 kg / meter / / / ≥ 27 kg / m / 18. 5 - 24. 9 kg/m/2

It is amazing that that from the entire document content, the model picked exactly the relevant numbers for the BMI, including the units.



Coronary Artery Disease (CAD)

TOPIC IMAGES (1) UPDATES

Overview and Recommendations
Background
Evaluation
Management

Overview and Recommendations

Background

- CAD refers to atherosclerotic narrowing of coronary arteries t
  early in the course of the disease but may lead to stable or un
  myocardial infarction with the progressive thickening or plaqu
  the coronary arteries.

Management

- Institute the following lifestyle modifications for all patients with CAD:
  - Limit daily intake of saturated fats (< 7%-10%) and trans-fatty acids (< 1%)
    recommendation).
  - Increase daily consumption of fresh fruits and vegetables (Strong recomm
  - Encourage regular aerobic physical activity with durations ranging from 2(
    60 minutes/day and recommended frequency ranging from ≥ 3 times/wee
    days/week (Strong recommendation).
  - Increase daily consumption of fresh fruits and vegetables
  - Encourage regular aerobic physical activity with durations
    60 minutes/day and recommended frequency ranging fro
    days/week (Strong recommendation).
  - Weight management is recommended to achieve and mai
    - body mass index (BMI) 18.5-24.9 kg/meter^2

Web Mining and Analytics, Content

- Web Crawling
- Web Scraping
- A Web Crawling Illustration
- Understanding Search Performance
- Using Shiny

Web Scraping and Crawling

- Almost 90% of the data in the world in 2017, has been created in the last two years alone.
- Our current output of data is roughly 2.5 quintillion bytes a day.
- As the world steadily becomes more connected with an ever-increasing number of electronic devices, this number is only set to grow over the coming years.

source: https://www.iflscience.com/technology/how-much-data-does-the-world-generate-every-minute/

Web is probably the easiest and most available platform widely used for information collection and analysis.

Data Collection from Web

- **Web Scraping**: The process of automatically collecting data from web pages or web resources is called web scrapping. This process is also called web knowledge extraction

- **Web Crawling**: The process of reading and storing all web pages of a site or number of sites is called web crawling. In other words, the process of web crawling is related to gathering pages from the web and indexing them to support a search engine.
  - The objective of crawling is quickly and efficiently gather as many useful web pages as possible, including the link structure that interconnects them.
  - The applications which is performing the web crawling are called **bot**. Nevertheless, bots are not limited only to web crawling, they can simulate human tasks as well.

Scraping vs Crawling

- **Web Scraping** is focusing on a single type of a web resource.

- **Web Crawling** is automatically reading and indexing all web pages of a web site. Note that a web site is composed of many web pages.
  - Web crawling is indexing web pages, and search engines use web crawling to index web pages.
  - web crawling = web spider = web robots
  - Web crawling examples: Collecting email addresses (Spammer), indexing web pages for fast access

Why Web Scrapping?

Web scrapping can read pages, that is hard for human to read and summarize.

Example: '*finding me the cheapest flight from Boston to San Diego.*'

Web Scrapping is so useful that some web sites provide their own **API** (Application Programable Interface) for scrapping their data, e.g., Twitter API, Instagram API, Youtube API, WikiPedia API, …

Many applications benefit from web scrapping

- **Market forecasting and market studies**, *e.g., scrapping online product review from Amazon, identifying public opinion about a corporation from twitter*.
- **Machine-language translation**, *e.g., using web text as a template to reconstruct a sentence correctly .*
- **Medical diagnostics** (retrieve and analyze data from news sites, translated texts, and health forums), *e.g., reading health forums to identify a symptom of a drug in large scale.*
- **Opinion mining from social/news media**, *e.g. How is Y (e.g., news media) opinion changes about subject X (e.g., a country)? How is public opinion about product Z.*
- **Epidemic propagation**, *e.g., influenza based on geolocation, amount of hate speech (tweets) during different times, …*

Some Available Dataset Examples

- **Crowd Fundings**
  - Kickstarter: https://webrobots.io/kickstarter-datasets/
  - Indiegogo: https://webrobots.io/indiegogo-dataset/

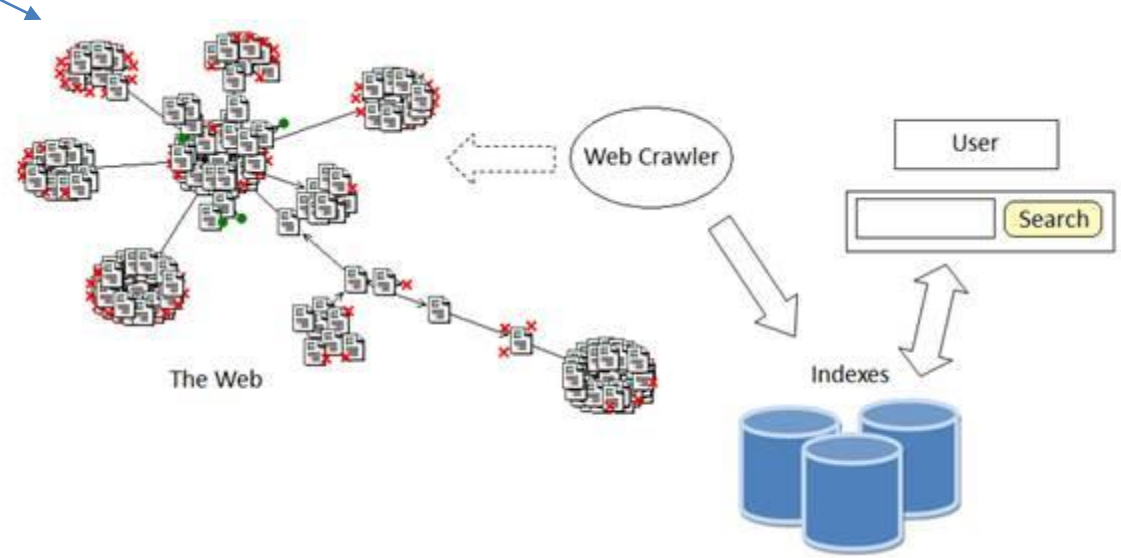
- **Academia**
  - DBLP: https://dblp.uni-trier.de/xml/
  - Pubmed: https://www.nlm.nih.gov/databases/download/pubmed_medline.html

- **Professional Community**
  - Stack Exchange: https://archive.org/details/stackexchange

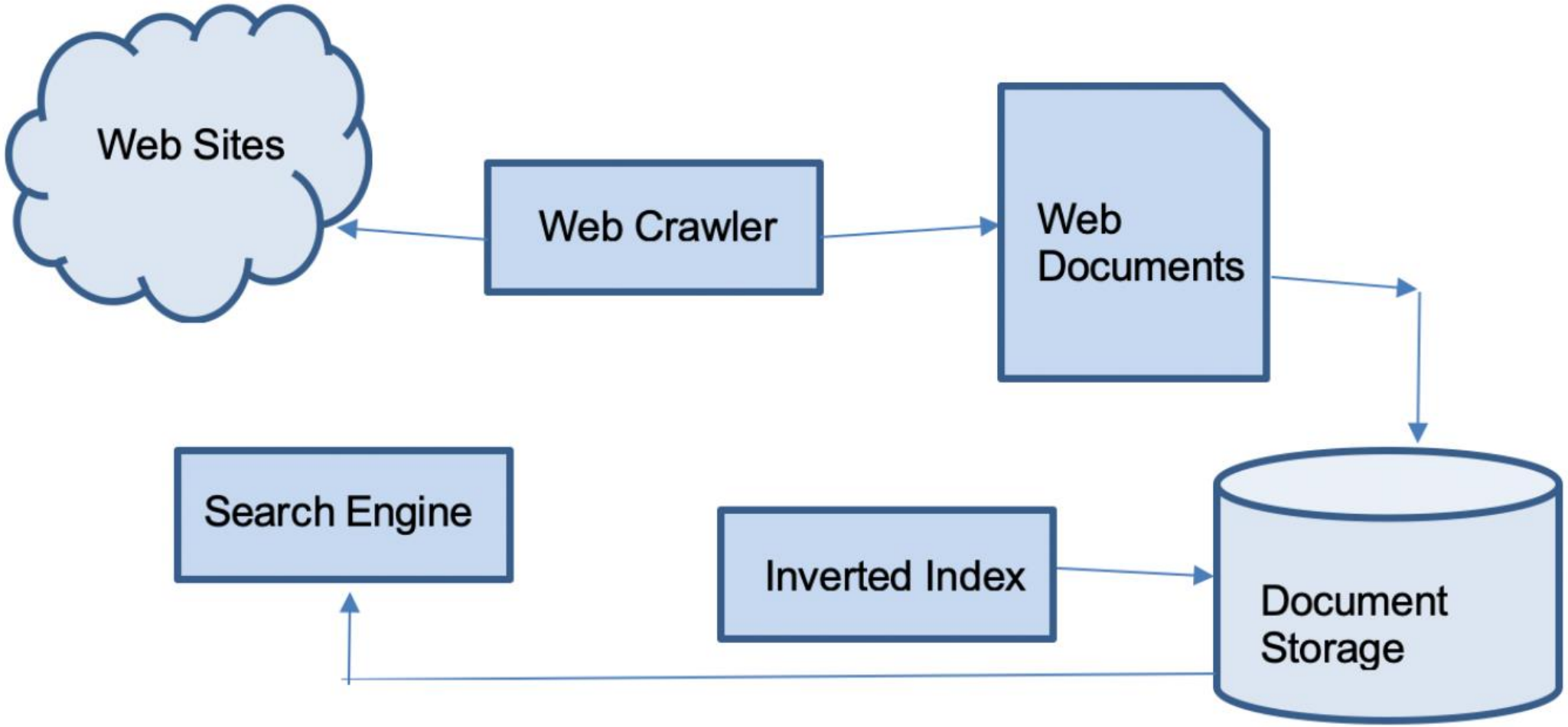
Components of a web search engine

- The techniques for web mining are similar to the ones used for text mining with the exception of the use of a search engine.

- The search engine has the following architecture
  - Content Aggregator (Crawling Subsystem, Google, Yahoo etc. search)
  - Indexing Subsystem
  - Search Interface
  - User (Content Consumer)



Search Engine Overview



Web Crawlers

- A web crawler fetches, analyses and files information from web servers.
- Web crawlers (sometimes referred to as a spider) can copy all the indexed pages they visit for quicker processing by a search engine.

- The basic operational steps of a hypertext crawler are
  - Begin with one or more URLs that constitute a seed set
  - Fetch the web page from the seed set
  - Parse the fetched web page to extract the text and the links
    - Extracted text is fed to a text indexer
    - Extracted links (URLs) are added to URLs whose corresponding pages have yet to be fetched by the crawler
  - The visited URLs are deleted from the seed set

- It is a recursive traversal of a web graph where each node is a URL.
- Multi-threaded design to process a large number of web pages quickly (a **fetch rate**).
  - A fetch rate of hundred pages each second will fetch a billion pages in 1-month long crawl.
  - This is a small fraction of the static Web at present (massively distributed parallel computing typically used).

Search engine indexing

- Web (Internet) indexing refers to various methods for indexing the contents of a website or of the Internet as a whole.
- Metadata web indexing - assigning keywords or phrases to web pages or web sites within a metadata tag field
  - So that the web page or web site can be retrieved with a search engine that is customized to the search in the keywords field.
- Web pages with frequent changes would require dynamic indexing.
- For very large data collections like the web, indexing has to be distributed over computer clusters with hundreds or thousands of machines.
- The basic steps in constructing an index - A term–document ID pairs (as described in text mining).
- Considering the large data collections, index implementations consists of
  - Partitioning by **terms**, also known as **global** index organization.
  - Partitioning by **documents** (more common), also known as **local** index organization.

WebSPHINX Crawler

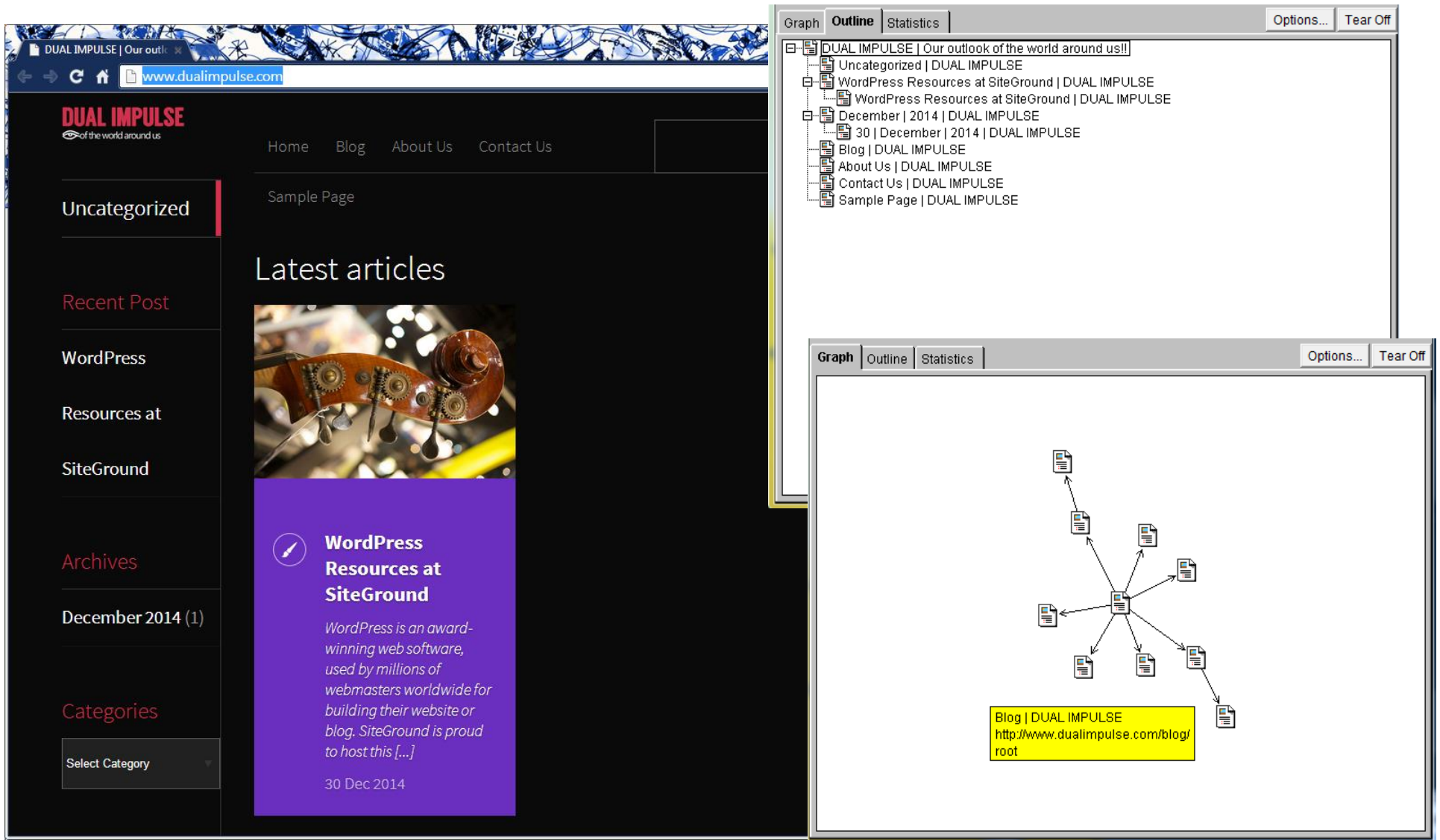
- WebSPHINX is a GUI tool and a Java class library that can be downloaded from:
  http://www.cs.cmu.edu/~rcm/websphinx/#download

- WebSPHINX – a web crawler that allows you to experiment with some basic crawls automatically, over a small part of the web (a single web site).
  - Open source
  - Intended for a personal use (not a professional)
  - Customizable
  - Visualizes the collection of the visited web pages as a graph.
  - It saves the visited web pages for offline browsing.
  - It also can extract pattern of text matching criteria from the collection of pages.

- **Note**: Make sure you crawl over the web sites that would not restrict your access after the first test crawl. Many web sites will restrict your access after crawling with WebSPHINX.

Try to maybe crawl over your classmate's web site from the Google Analytics Project.

- Here is crawl of the website http://www.dualimpulse.com/

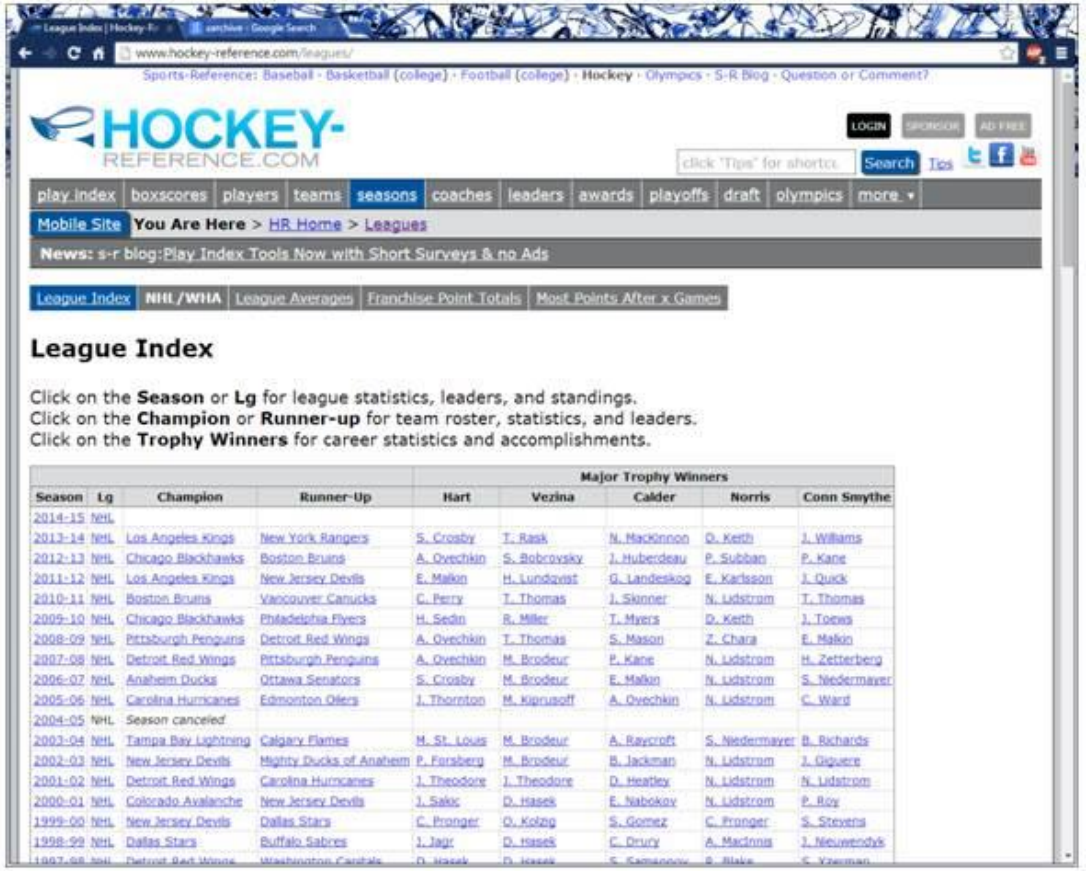


WebSPHINX Illustration

The following URL "http://www.hockey-reference.com/leagues/"

contains the hockey league reference information on the NHL seasons going back to 1917.

This site has a lot of interlinked web pages containing relevant and overlapping information, thus making it an illustrative test example.

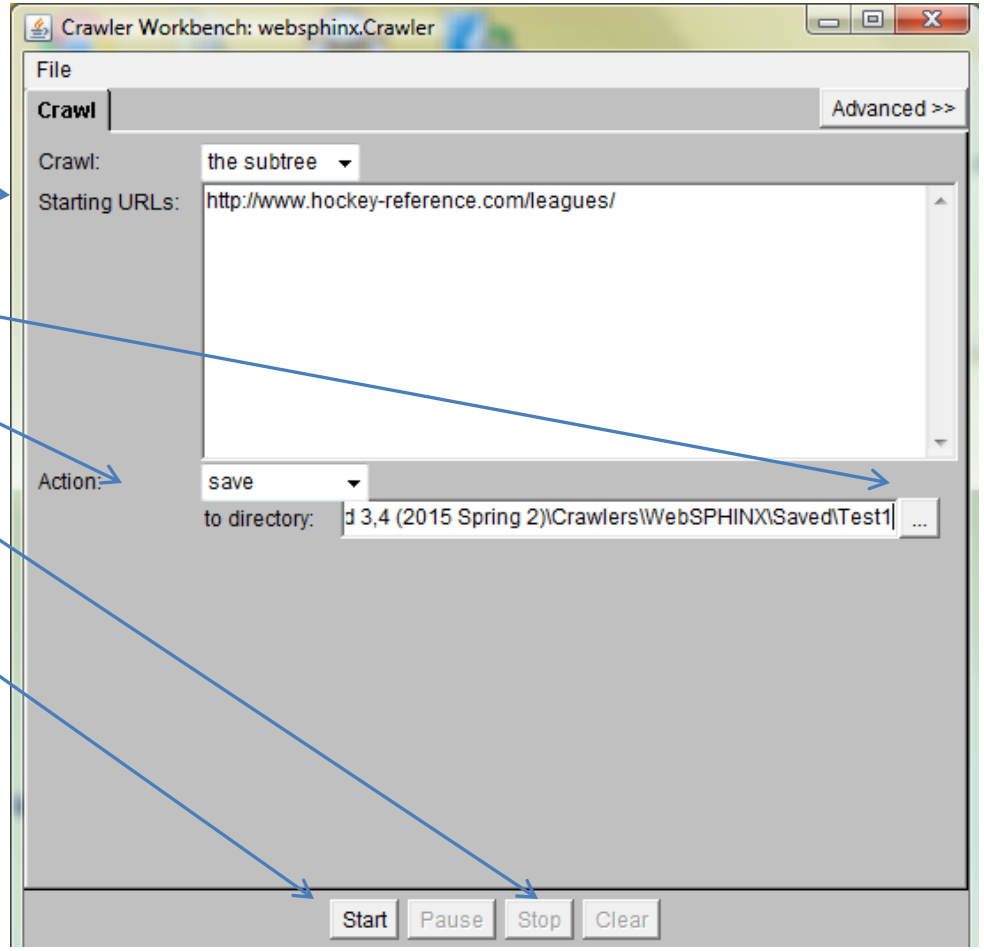


Running WebSPHINX

Choosing the crawl options for WebSPHINX.

- Type the URL
- Specify Action - save
- Enter location to save visited HTML files
- Then click start

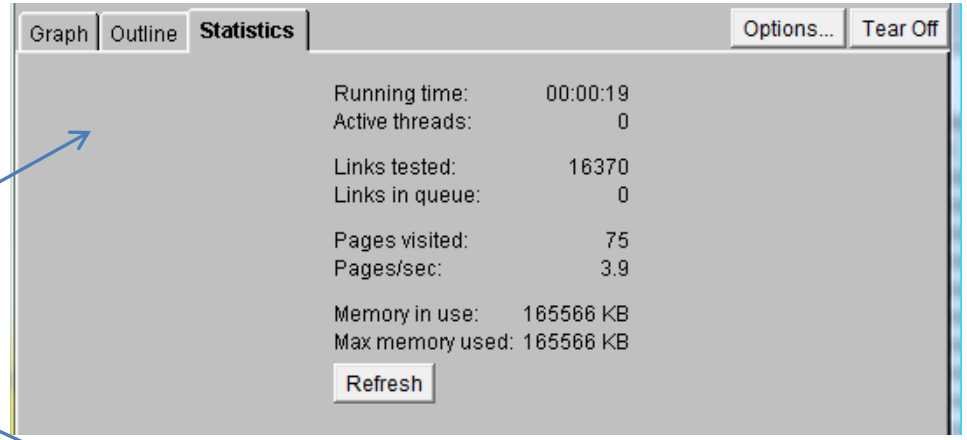
After some time you can stop the crawl

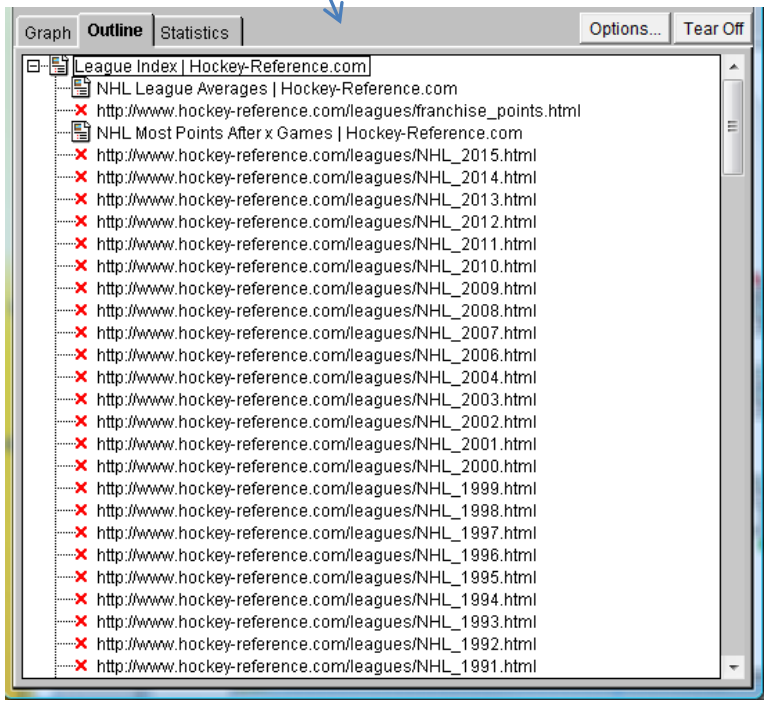


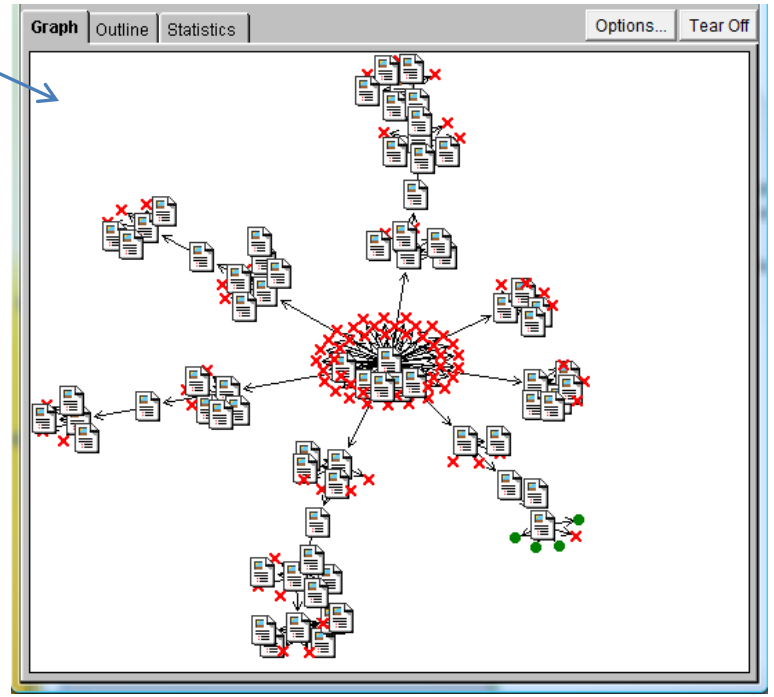
Running WebSPHINX

The web pages explored are visualized

- As a graph
- As a list
- The statistics about the crawl







Lab Project: WebSPHINX Crawler

- Use WebSPHINX to crawl over one of the web sites (yours or one of your classmate's) created for the Google Analytics project.
- **Note**: Make sure you crawl over the web sites that would not restrict your access after the first test crawl. Many web sites will restrict your access after crawling with WebSPHINX.
- Present your crawl results in a slide similar to the illustrations on the previous slides.
- Make sure to include the URL and the screenshots of the web pages as a graph.

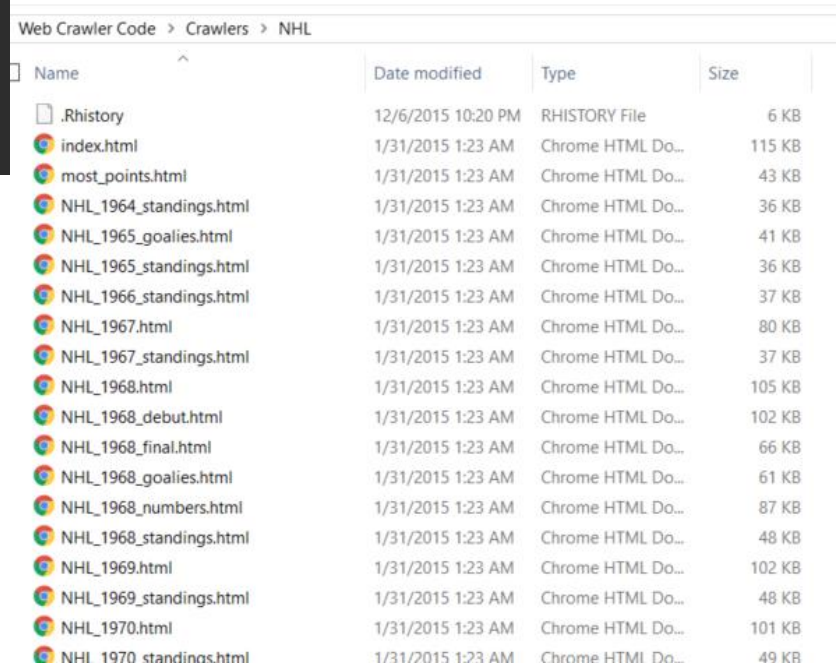
Analyzing WebSPHINX results with R

The following code illustrates how the saved data from WebSPHINX crawl can be analyzed with R.

```r
# Module 4 Code

### --- Example 1: Analyze WebSPHINX results. --------
library(XML)
dir <-file.path(paste0('.\\Crawlers\\','NHL\\'))
HTML.dataset <- list.files(dir,pattern ="html") # List of all saved HTML files @ location "dir"

# Function to strip the table data from the HTML files
sieve.HTML <- function(URL) {
  table <- readHTMLTable(URL) # Read HTML table into a list
}

temp.HTML.text <- lapply(as.list( paste0(dir,HTML.dataset)),function(x) sieve.HTML(x)) # Get all the text from the saved HTMLs

query <- "Boston Bruins"
temp <- grep(query, temp.HTML.text[[1]][[1]]$Champion)
# [1]  5 51 53 82 84 94
temp.HTML.text[[1]][[1]]$Season[temp]

```

This code gives the seasons when "Boston Bruins" were champions.

Saved data from a crawl over the hockey league website resides in a folder Crawlers/NHL/



Simple Web Pages Analysis in R

- After the crawl, the saved web pages can be analyzed for content.
- Example below illustrates how the R code is used
  - To strip the table data from the HTML files.
  - Assumes the data was saved at location specified by object "dir".

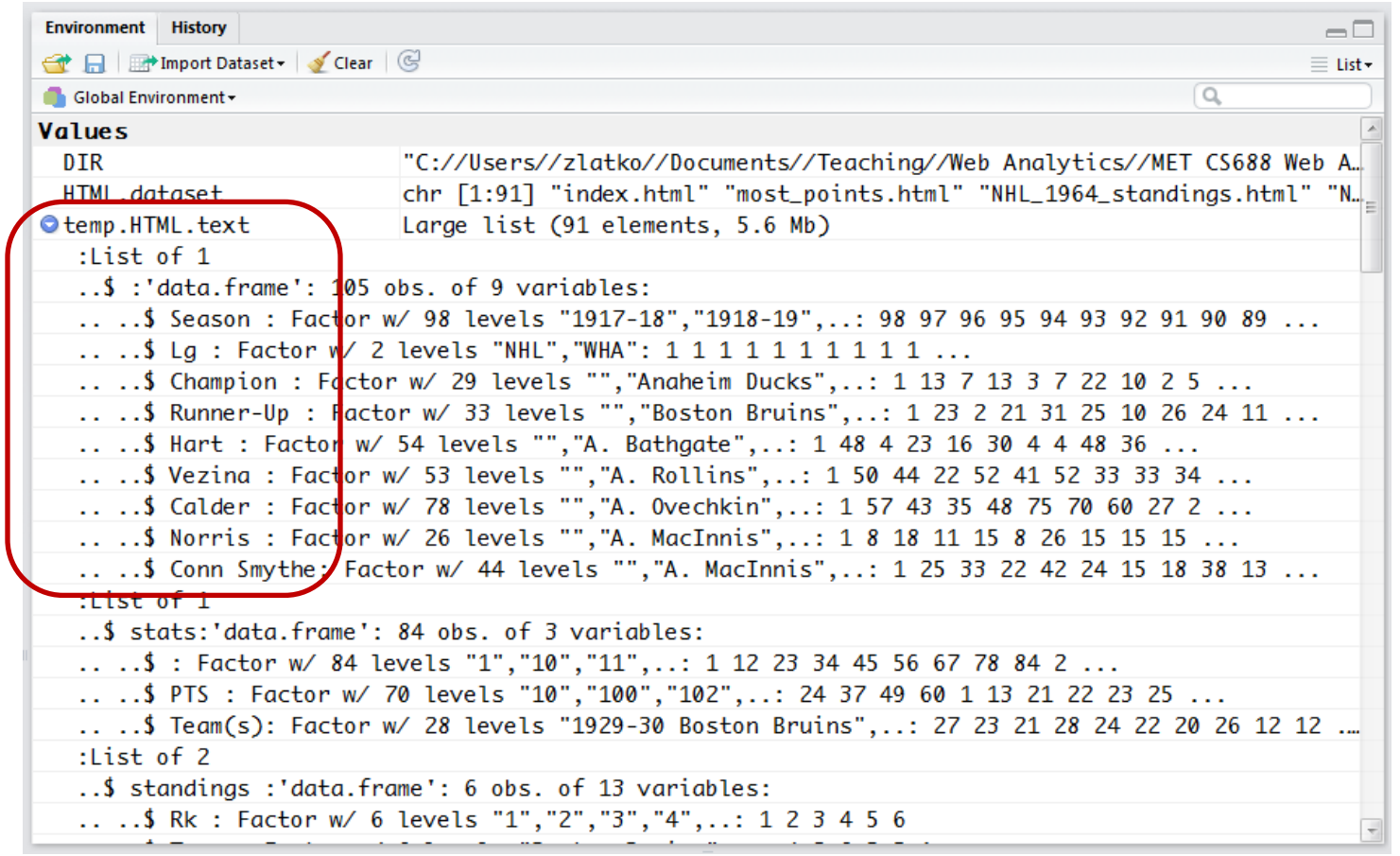
```
# Example: Analyze WebSPHINX results
library(XML)
HTML.dataset <- list.files(dir,pattern ="html") # List of all saved HTML files @ location "dir"

# Function to strip the table data from the HTML files
sieve.HTML <- function(URL) {
  table <- readHTMLTable(URL) # Read HTML table into a list
}
temp.HTML.text <- lapply(HTML.dataset,function(x) sieve.HTML(x)) # Get all the text from the saved HTMLs
```

  - HTML.dataset contains the names of all of the crawled and saved web pages.
  - For each saved HTML file the function sieve.HTML() is called through lapply().

Simple Web Pages Analysis in R

- The object temp.HTML.text is a list of many (in this case 91) data tables.
- For the Index.html page (the first data frame in the object temp.HTML.text) You can see the column names and access the data for example in the “Seasons” column as:

  temp.HTML.text[[1]][[1]]$Season



Simple Web Pages Analysis in R

- The first (*index.html*) file contains the list of all of the NHL champions since 1917
- Note the column titles in the *index.html* page. The third is “Champions”
- You can check that in the data table object temp.HTML.text also.

```
temp.HTML.text[[1]][[1]]
```

- If we search for "Boston Bruins" the column “Champions”

```
> query <- "Boston Bruins"
> temp <- grep(query, temp.HTML.text[[1]][[1]]$Champion)
[1]  5 51 53 82 84 94
```

- We get the indices where the query appears
- As an example we can get the corresponding seasons when the Bruins were champions.

```
temp.HTML.text[[1]][[1]]$Season[temp]
2010-11
1971-72
1969-70
1940-41
1938-39
1928-29
```

League Index

Click on the **Season** or **Lg** for league statistics,
Click on the **Champion** or **Runner-up** for team
Click on the **Trophy Winners** for career statist

Season	Lg	Champion	Runner-Up
2015-16	NHL		
2014-15	NHL	Chicago Blackhawks	Tampa Bay Lightning
2013-14	NHL	Los Angeles Kings	New York Rangers
2012-13	NHL	Chicago Blackhawks	Boston Bruins
2011-12	NHL	Los Angeles Kings	New Jersey Devils
2010-11	NHL	Boston Bruins	Vancouver Canucks
2009-10	NHL	Chicago Blackhawks	Philadelphia Flyers
2008-09	NHL	Pittsburgh Penguins	Detroit Red Wings
2007-08	NHL	Detroit Red Wings	Pittsburgh Penguins
2006-07	NHL	Anaheim Ducks	Ottawa Senators
2005-06	NHL	Carolina Hurricanes	Edmonton Oilers
2004-05	NHL	*Season canceled*	

About EDGAR

EDGAR, the Electronic Data Gathering, Analysis, and Retrieval system, performs automated collection, validation, indexing, acceptance, and forwarding of submissions by companies and others who are required by law to file forms with the U.S. Securities and Exchange Commission (the "SEC"). The database is freely available to the public via the Internet (Web or FTP).

Visit : http://www.sec.gov/cgi-bin/browse-edgar?company=frontline&owner=exclude&action=getcompany

Example: Retrieving Financial Data from EDGAR.

1. Familiarize yourself with the HTML structure of the site and the relevant pages you would like to scrape.
2. Use library(rvest)
3. Form a query: SearchQuery <- "frontline“
4. Use string concatenation (paste0()) to form the above link with SearchQuery instead of frontline
5. Create a user defined function to strip the data from the URL
   you can use some of “rvest” functions such as “read_html()”, “html_nodes()”, “html_text()”
6. If needed, call the user defined function with lapply()

Note: Scraping web pages requires great deal of programming skils.

Scraping Financial Data from EDGAR

EDGAR stands for the Electronic Data Gathering, Analysis, and Retrieval system. It is a US. Securities and Exchange Commission website.

http://www.sec.gov/edgar/searchedgar/companysearch.html

This website performs automated collection, validation, indexing, acceptance, and forwarding of submissions by companies and others who are required by law to file forms with the U.S. Securities and Exchange Commission (the "SEC"). ").

The database is freely available to the public via the Internet (Web or FTP). There are Other countries' equivalents to EDGAR such as SEDAR in Canada for example or Companies House, United Kingdom.

See posted code examples using the package “edgarWebR”.

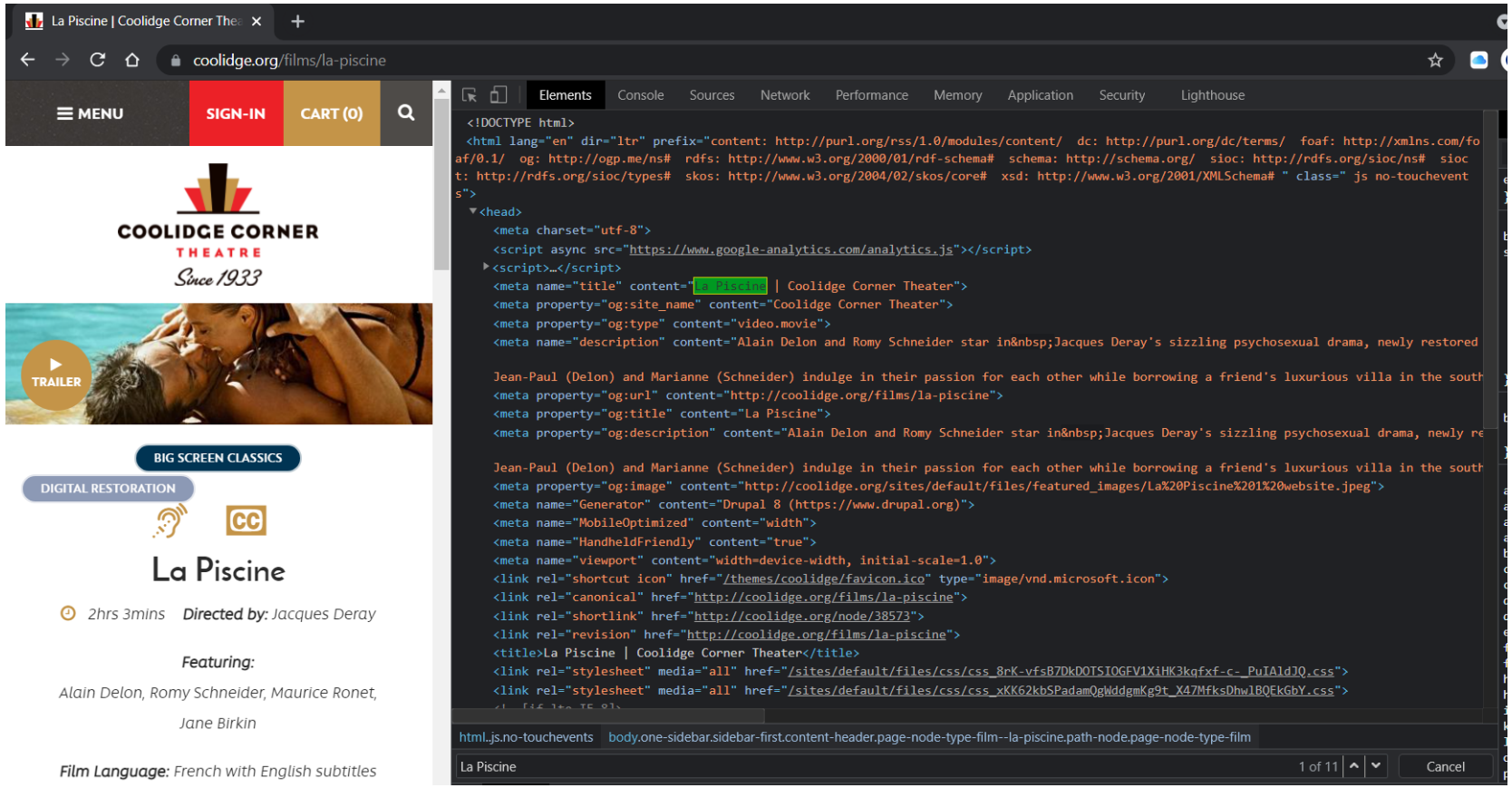
NOTE: You need to execute EDGAR functions several time to actually get the data!

Note: Scraping web pages requires great deal of programming skills.

Retrieving Movie Showtimes

Another relatively simple example of webpage (HTML) content scraping.

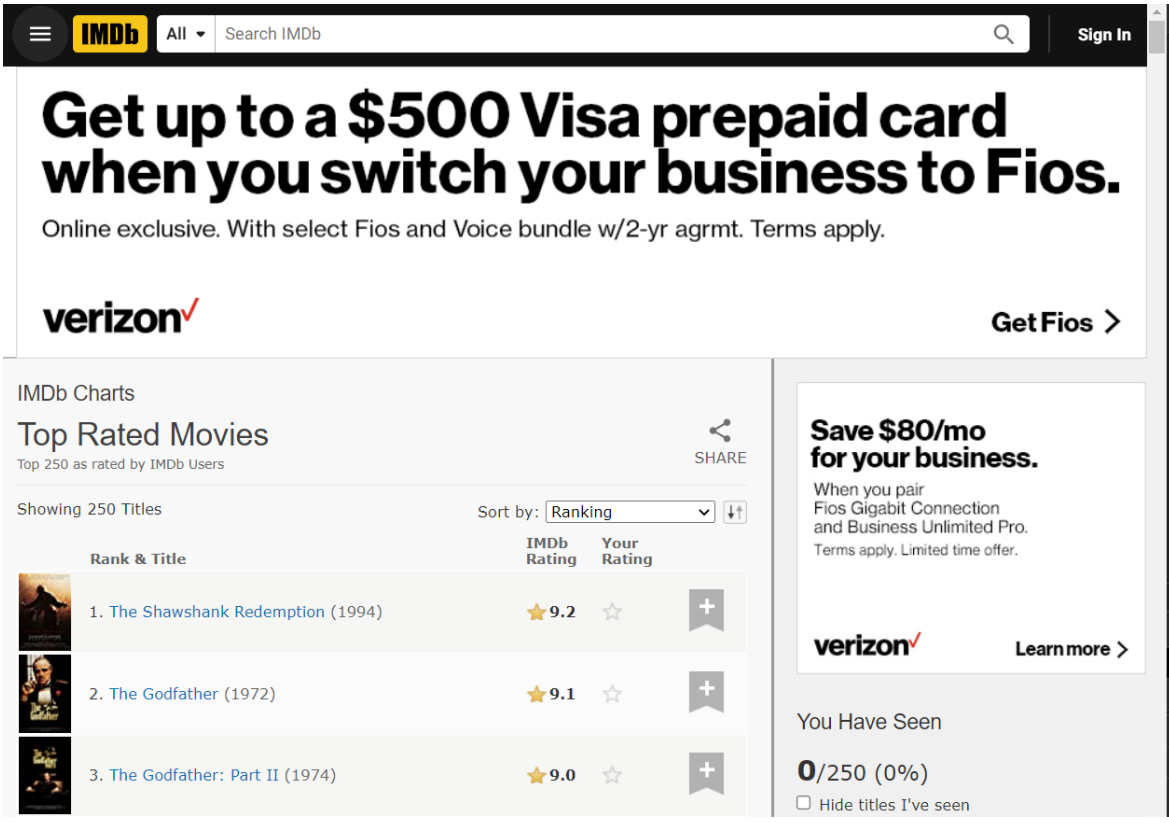
- Note how the relevant tags need to be addressed.
- It assumes familiarity to the web page structure.

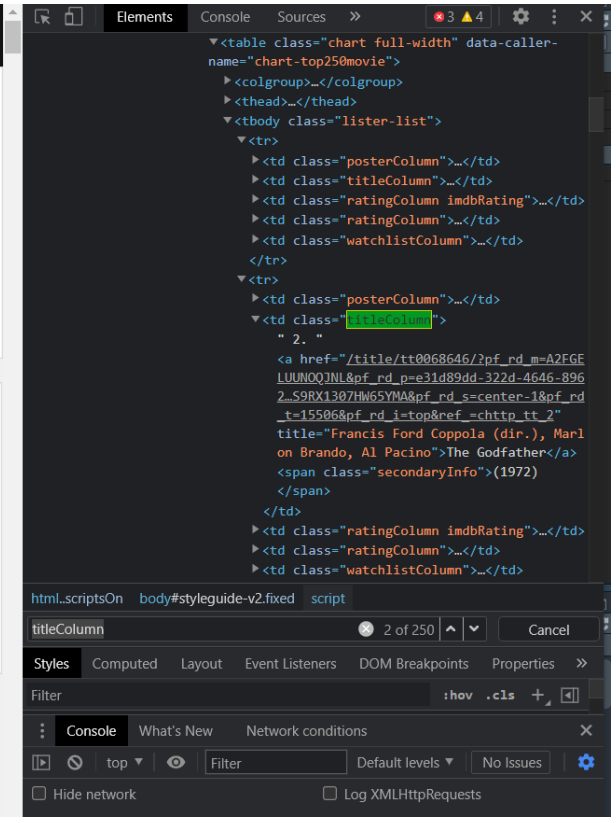



```
### --- Example 3: Access Current Movies Showtimes. --------
rm(list=ls()); cat("\014") # clear all
library("rvest")
movies <- read_html("https://coolidge.org/")
html_nodes(movies,'.film-card__link') %>% html_text()
movie.nodes <- html_nodes(movies,'.film-card__link') %>% html_text()
sapply(html_attrs(movie.nodes),`[[`,'title')
```

Scraping IMDB site for Most Popular Movies

Note that web sites frequently change so you would need to look at the web page source (ctrl+shift+i) in Chrome browser to explore which html tags to reference in your code.






```
# Scraping IMDB site for Most Popular Movies
url = "http://www.imdb.com/chart/top?ref_=nv_wl_img_3"
page = read_html(url)
movie.nodes <- html_nodes(page,'.titleColumn a')
movie.name  <- html_text(movie.nodes)
movie.cast <- sapply(html_attrs(movie.nodes),`[[`,'title')
```

```
> head(movie.name, 15)
 [1] "The Shawshank Redemption"
 [2] "The Godfather"
 [3] "The Godfather: Part II"
 [4] "The Dark Knight"
 [5] "12 Angry Men"
 [6] "Schindler's List"
 [7] "The Lord of the Rings: The Return of the King"
 [8] "Pulp Fiction"
 [9] "The Good, the Bad and the Ugly"
[10] "The Lord of the Rings: The Fellowship of the Ring"
[11] "Fight Club"
[12] "Forrest Gump"
[13] "Inception"
[14] "The Lord of the Rings: The Two Towers"
[15] "Star Wars: Episode V - The Empire Strikes Back"
>
```

Accessing/Modifying Information in XML Files

- A
- A
- The XML file is a standard way of keeping information for many professional applications.
- Note the code for this topic contains the commented XML used there.
- To use the code below, you need to uncomment the XML part and save it with a text editor as an XML file.

```r
### --- Example 4: Access XML File. --------
## sequencing.xml file content
# <data>
#   <sequence id = "ancestralSequence">
#   <taxon id="test1">Taxon1
# </taxon>
#   GCAGTTGACACCCTT
# </sequence>
#   <sequence id = "currentSequence">
#   <taxon id="test2">Taxon2
# </taxon>
#   GACGGCGCGGAccAG
# </sequence>
#   </data>

pth <- file.path("c:", "Users", "ZlatkoFCX","Documents","My Files","Tea
library(XML)
# read XML File located in folder "pth"
x = xmlParse(file.path(pth,"sequencing.xml"))

# returns a *list* of text nodes under "sequence" tag
nodeSet = xpathApply(x,"//sequence/text()")

# loop over the list returned, and get and modify the node value:
zz<-sapply(nodeSet,function(G){
  text = paste("Ggg",xmlValue(G),"CCaaTT",sep="")
  text = gsub("[^A-Z]","",text)
  xmlValue(G) = text
})

```

sequencing.xml

```xml
<data>
  <sequence id = "ancestralSequence">
  <taxon id="test1">Taxon1
</taxon>
  GCAGTTGACACCCTT
</sequence>
  <sequence id = "currentSequence">
  <taxon id="test2">Taxon2
</taxon>
  GACGGCGCGGACCAG
</sequence>
  </data>

```

A

- A