



MET CS688 C1

WEB ANALYTICS AND MINING

ZLATKO VASILKOSKI

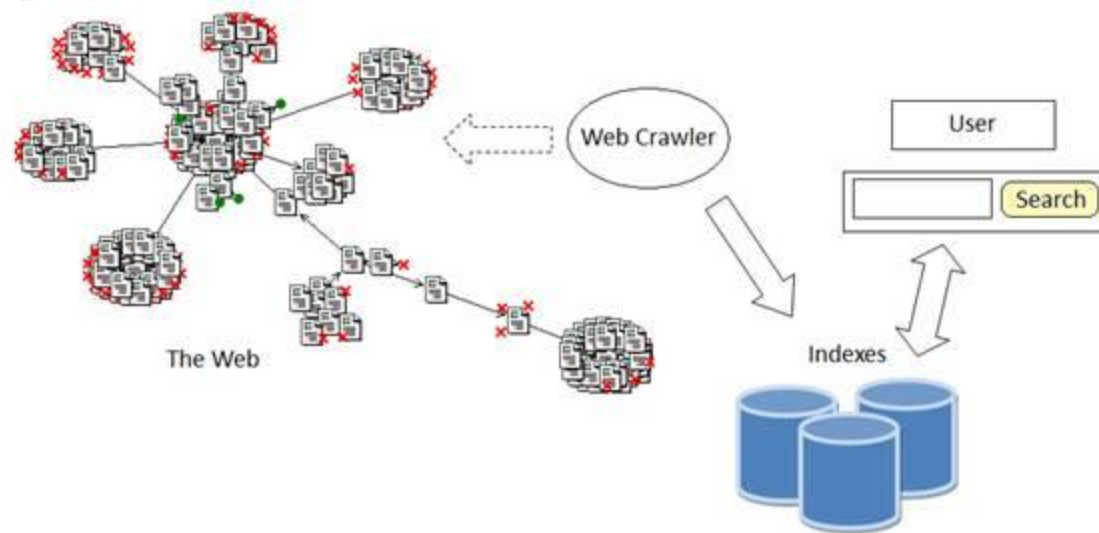
WEB MINING

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.

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)



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.

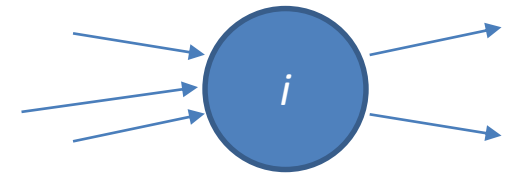
- Although much of it is duplication (30-40%)
- Best estimate of “unique” static HTML pages comes from search engine claims.
- Google claimed and Yahoo claimed tens of billion.
- Google recently announced that their index contains 1 trillion pages.

The web as a graph

- Directed graph
 - Pages - nodes,
 - hyperlinks - edges
- High linkage 10-20 links/page on average
 - The links between the web pages are not randomly distributed
 - Power-law degree distribution (see Module 4 Discussion)

Power-law degree distribution

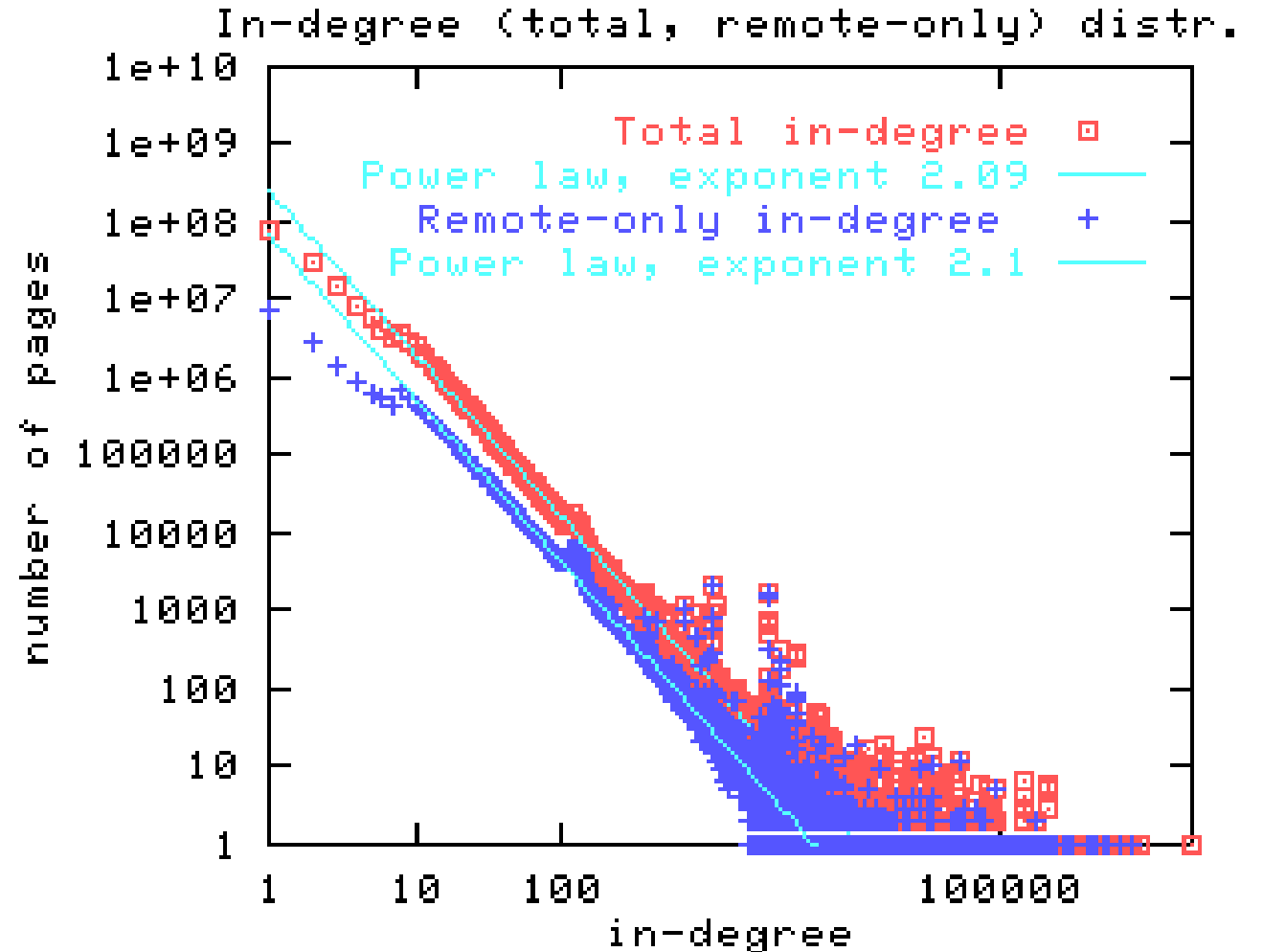
- Assigning properties to the edges of a graph, such as a web page endorsements turns the graph into a network.
- The empirical study of networks has played an important role from biological and social to telecommunication, computer and web networks.
- In the case of the Web, the links between the web pages are not randomly distributed.
- They follow a power law distribution based on the in-node degree (i).
- The total number of web pages with in-degree i , almost as a natural law, is proportional to $i^{-\beta}$, where $\beta \approx 2.1$, strongly indicating some aspects about the spontaneous association structure of the Web.



Power-law degree distribution

- The total number of web pages vs. in-degree i

- Source: Broder et al, 2000



Approaches to Analyzing 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.

Sliding Windows Big Data Example

Model of stream processing

q w e r t y u i o p **a s d f g h** j k l z x c v b n m

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

Big Data model extension (lots of online retailer examples)

- N is 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

“Alphabet” Simplification

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

Sliding Windows Big Data Example

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 $N=1\text{billion}$.

Note: We **can not** 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”.
 - **Question:** How many 1’s are in the last N bits?
 - **Question:** What are the assumptions on the streaming of “0” and “1” in this approach?

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"

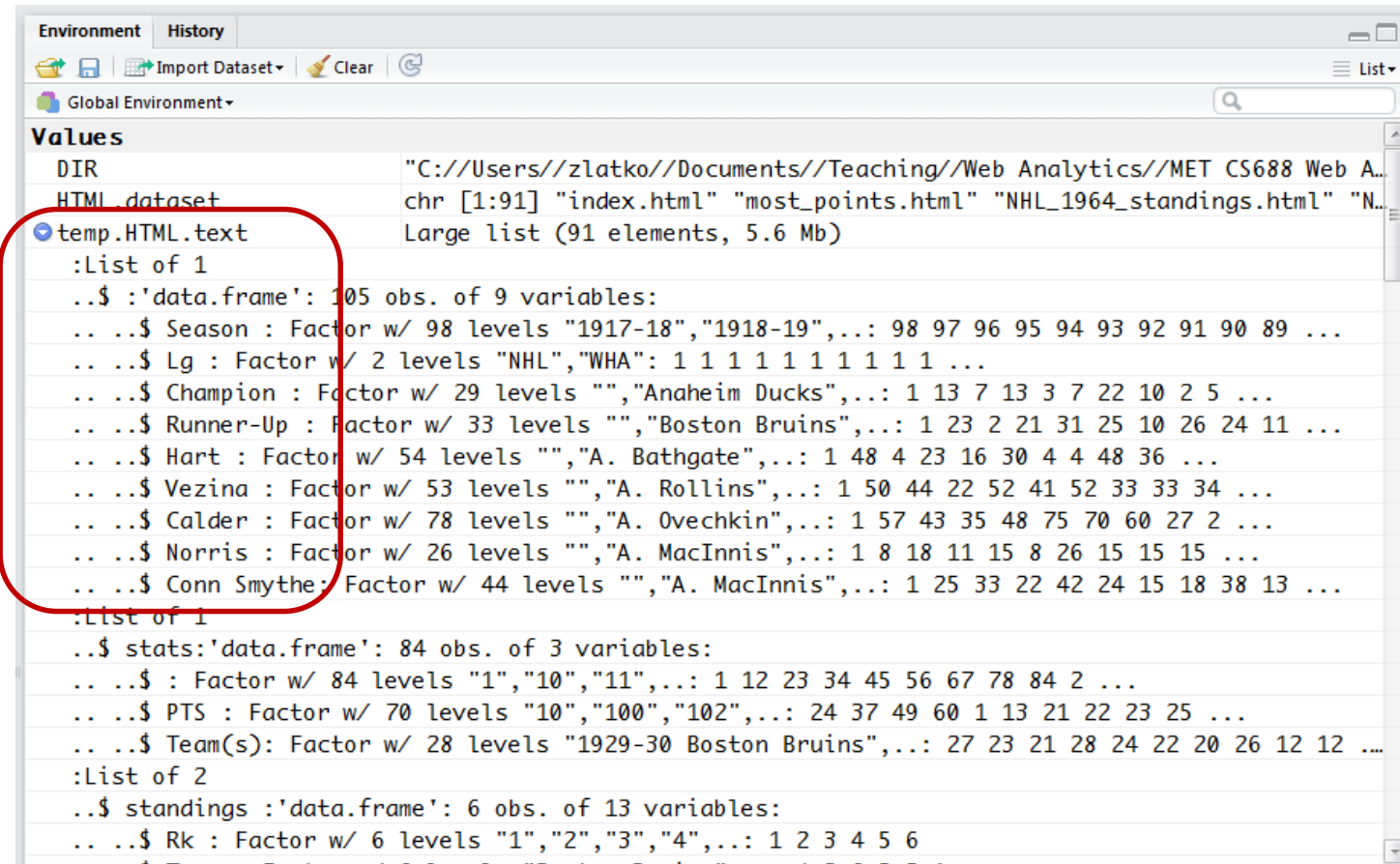
# Function to strip the table data from the HTML files
sieve.HTML <- function(URL) {
  table <- readHTMLTable(URL) # Read HTML table into a list
}
temp.HTML.text <- lapply(HTML.dataset,function(x) sieve.HTML(x)) # Get all the text from the saved HTMLs
```

- `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
```



The screenshot shows the R Studio Environment pane with the 'temp.HTML.text' object selected. The object is a list of 91 elements, totaling 5.6 Mb. The first element is a data frame with 105 observations and 9 variables. The variables are: Season (Factor w/ 98 levels), Lg (Factor w/ 2 levels), Champion (Factor w/ 29 levels), Runner-Up (Factor w/ 33 levels), Hart (Factor w/ 54 levels), Vezina (Factor w/ 53 levels), Calder (Factor w/ 78 levels), Norris (Factor w/ 26 levels), and Conn Smythe (Factor w/ 44 levels). The second element is a data frame with 84 observations and 3 variables: stats (Factor w/ 84 levels), PTS (Factor w/ 70 levels), and Team(s) (Factor w/ 28 levels). The third element is a data frame with 6 observations and 13 variables: standings (Factor w/ 6 levels).

```
Environment History
Global Environment
Values
DIR "C://Users//zlatko//Documents//Teaching//Web Analytics//MET CS688 Web A...
HTML_dataset chr [1:91] "index.html" "most_points.html" "NHL_1964_standings.html" "N...
temp.HTML.text Large list (91 elements, 5.6 Mb)
:List of 1
..$ : 'data.frame': 105 obs. of 9 variables:
.. ..$ Season : Factor w/ 98 levels "1917-18","1918-19",...: 98 97 96 95 94 93 92 91 90 89 ...
.. ..$ Lg : Factor w/ 2 levels "NHL","WHA": 1 1 1 1 1 1 1 1 1 1 ...
.. ..$ Champion : Factor w/ 29 levels "", "Anaheim Ducks",...: 1 13 7 13 3 7 22 10 2 5 ...
.. ..$ Runner-Up : Factor w/ 33 levels "", "Boston Bruins",...: 1 23 2 21 31 25 10 26 24 11 ...
.. ..$ Hart : Factor w/ 54 levels "", "A. Bathgate",...: 1 48 4 23 16 30 4 4 48 36 ...
.. ..$ Vezina : Factor w/ 53 levels "", "A. Rollins",...: 1 50 44 22 52 41 52 33 33 34 ...
.. ..$ Calder : Factor w/ 78 levels "", "A. Ovechkin",...: 1 57 43 35 48 75 70 60 27 2 ...
.. ..$ Norris : Factor w/ 26 levels "", "A. MacInnis",...: 1 8 18 11 15 8 26 15 15 15 ...
.. ..$ Conn Smythe: Factor w/ 44 levels "", "A. MacInnis",...: 1 25 33 22 42 24 15 18 38 13 ...
:List of 1
..$ stats: 'data.frame': 84 obs. of 3 variables:
.. ..$ : Factor w/ 84 levels "1","10","11",...: 1 12 23 34 45 56 67 78 84 2 ...
.. ..$ PTS : Factor w/ 70 levels "10","100","102",...: 24 37 49 60 1 13 21 22 23 25 ...
.. ..$ Team(s): Factor w/ 28 levels "1929-30 Boston Bruins",...: 27 23 21 28 24 22 20 26 12 12 ...
:List of 2
..$ standings : 'data.frame': 6 obs. of 13 variables:
.. ..$ Rk : Factor w/ 6 levels "1","2","3","4",...: 1 2 3 4 5 6
```

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 skills.

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.

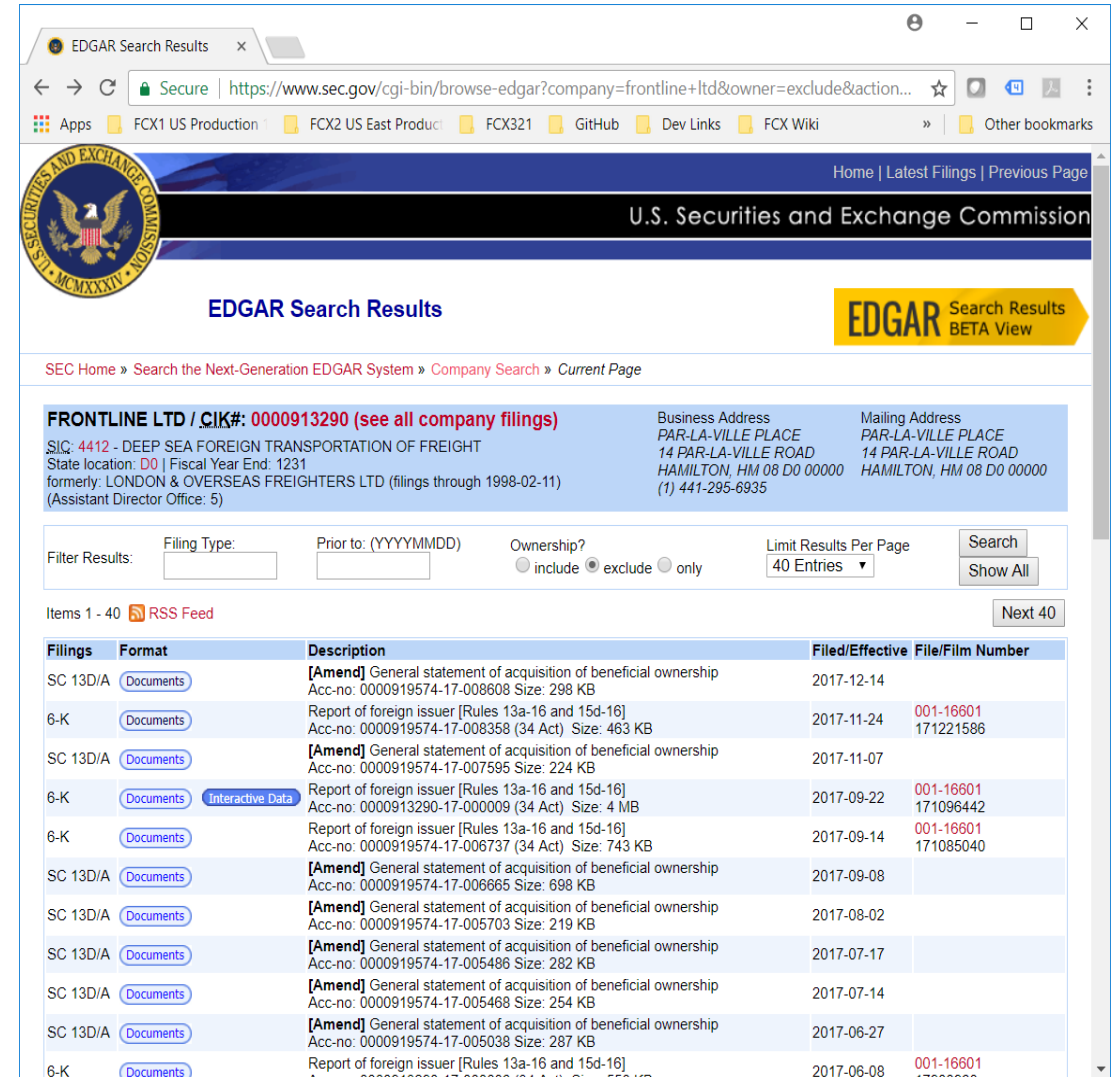
<http://www.sec.gov/edgar/searchedgar/companysearch.html>

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: Scraping web pages requires great deal of programming skills.

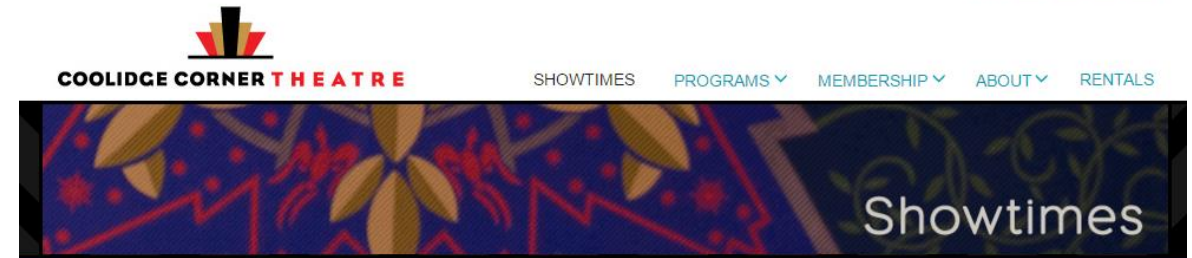


The screenshot shows the EDGAR Search Results page for Frontline Ltd. (CIK# 0000913290). The page displays company information, including business and mailing addresses, and a list of filings. The filings table includes columns for Filings, Format, Description, Filed/Effective, and File/Film Number.

Filings	Format	Description	Filed/Effective	File/Film Number
SC 13D/A	Documents	[Amend] General statement of acquisition of beneficial ownership Acc-no: 0000919574-17-008608 Size: 298 KB	2017-12-14	
6-K	Documents	Report of foreign issuer [Rules 13a-16 and 15d-16] Acc-no: 0000919574-17-008358 (34 Act) Size: 463 KB	2017-11-24	001-16601 171221586
SC 13D/A	Documents	[Amend] General statement of acquisition of beneficial ownership Acc-no: 0000919574-17-007595 Size: 224 KB	2017-11-07	
6-K	Documents	Report of foreign issuer [Rules 13a-16 and 15d-16] Acc-no: 0000913290-17-000009 (34 Act) Size: 4 MB	2017-09-22	001-16601 171096442
6-K	Documents	Report of foreign issuer [Rules 13a-16 and 15d-16] Acc-no: 0000919574-17-006737 (34 Act) Size: 743 KB	2017-09-14	001-16601 171085040
SC 13D/A	Documents	[Amend] General statement of acquisition of beneficial ownership Acc-no: 0000919574-17-006665 Size: 698 KB	2017-09-08	
SC 13D/A	Documents	[Amend] General statement of acquisition of beneficial ownership Acc-no: 0000919574-17-005703 Size: 219 KB	2017-08-02	
SC 13D/A	Documents	[Amend] General statement of acquisition of beneficial ownership Acc-no: 0000919574-17-005486 Size: 282 KB	2017-07-17	
SC 13D/A	Documents	[Amend] General statement of acquisition of beneficial ownership Acc-no: 0000919574-17-005468 Size: 254 KB	2017-07-14	
SC 13D/A	Documents	[Amend] General statement of acquisition of beneficial ownership Acc-no: 0000919574-17-005038 Size: 287 KB	2017-06-27	
6-K	Documents	Report of foreign issuer [Rules 13a-16 and 15d-16] Acc-no: 0000913290-17-000006 (34 Act) Size: 550 KB	2017-06-08	001-16601 17000060

Retrieving Movie Showtimes

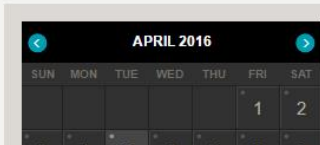
- Another relatively simple example of webpage (HTML) content scraping.
- Note how the relevant tags need to be addressed.
- It assumes familiarity to the page structure.



VIEW BY: Date

Today
This Week
Full Schedule

or select a date below



Tuesday April 5, 2016



Eye in the Sky

1hr 42mins

NEW RELEASES The commander is in England. The drone pilot is in America. The terrorist is in Kenya. And the authority to strike is up in the air. [MORE](#)

7:00pm 9:30pm



Hello, My Name is Doris

1hr 35mins

NEW RELEASES She's not ready to act her age. [MORE](#)

6:45pm 9:00pm

Midnight Special

```
47 ▾ ### --- Example 3: Access Current Movies Showtimes. -----
48 library("rvest")
49 movies <- read_html("http://www.coolidge.org/showtimes/")
50 titles <- html_nodes(movies,
51 "#view-id-external_showtime_date_browser-page div.film-event-title")
52 html_text(titles) # Display Current Movies
53 # [1] "Eye in the Sky" "Hello, My Name is Doris" "Midnight Special" "The Clan"
54
```

Accessing/Modifying Information in XML Files

- The XML file is a standard way of keeping information for many professional applications.
- Note the module 4 code contains the commented XML used there.
- You need to uncomment the XML part and save it as an XML file.

```
56 ▶ ### --- Example 4: Access XML File. -----
57 ## sequencing.xml file content
58 # <data>
59 #   <sequence id = "ancestralSequence">
60 #     <taxon id="test1">Taxon1
61 #   </taxon>
62 #     GCAGTTGACACCCTT
63 #   </sequence>
64 #   <sequence id = "currentSequence">
65 #     <taxon id="test2">Taxon2
66 #   </taxon>
67 #     GACGGCGCGGACcAG
68 #   </sequence>
69 # </data>
70
71 pth <- file.path("c:", "Users", "ZlatkoFCX", "Documents", "My Files", "Tea
72 library(XML)
73 # read XML File located in folder "pth"
74 x = xmlParse(file.path(pth, "sequencing.xml"))
75
76 # returns a *list* of text nodes under "sequence" tag
77 nodeSet = xpathApply(x, "//sequence/text()")
78
79 # loop over the list returned, and get and modify the node value:
80 zz<-sapply(nodeSet, function(G){
81   text = paste("Ggg", xmlValue(G), "CCaaTT", sep="")
82   text = gsub("[^A-Z]", "", text)
83   xmlValue(G) = text
84 })
85
```

```
sequencing.xml
1  <data>
2    <sequence id = "ancestralSequence">
3      <taxon id="test1">Taxon1
4    </taxon>
5      GCAGTTGACACCCTT
6    </sequence>
7    <sequence id = "currentSequence">
8      <taxon id="test2">Taxon2
9    </taxon>
10     GACGGCGCGGACCAG
11  </sequence>
12  </data>
13
```

Web sources Analysis in R

- The *tm.plugin.webmining* package is an open source framework for web mining applications.
- Facilitates the retrieval of textual data from the web.
 - In particular is implemented through WebSources (analogous to Source in “*tm*”)
- Source Name:
 - GoogleBlogSearchSource <http://www.google.com/blogsearch> - RSS
 - GoogleFinanceSource <http://www.google.com/finance> - RSS
 - GoogleNewsSource <http://news.google.com> - RSS
 - NYTimesSource <http://api.nytimes.com> - JSON
 - ReutersNewsSource <http://www.reuters.com/tools/rss> - ATOM
 - YahooFinanceSource <http://finance.yahoo.com> - RSS
 - YahooInplaySource <http://finance.yahoo.com/marketupdate/inplay> - HTML
 - YahooNewsSource <http://news.search.yahoo.com/rss> - RSS
- For corpus construction, *tm.plugin.webmining* uses a function call
 - WebCorpus() (analogous to *tm*'s Corpus() function)
- Once obtained the Corpus can be explored using *tm*'s text mining capabilities.

Note: To use this package you need R and Java to have matching architectures.

Exercise: Web Mining News

- Install needed packages.
- Perform “GoogleNewsSource” or “YahooNewsSource” search for a user specified query.
- This is illustrated on the next few slides.

“GoogleNewsSource” example

An illustration of using the *tm.plugin.webmining* package:

1. **WebSource** downloads the meta data from the feed.
 2. **WebCorpus** extracts the main content from the feed item.
- Let’s try the search for
 - **Query**: “Web Analytics”
 - **WebSource**: “GoogleNewsSource” RSS feed (or one of the listed on prev. slide)
 - Get the corpus:

```
result <- WebCorpus(GoogleNewsSource("Web Analytics"))
```
 - The data feed from Google News is pre-parsed with the function `GoogleNewsSource()` – a `WebSource` object .
 - `WebCorpus()` extracts the meta data from the `WebSource` object
 - downloads and extracts the actual main content of the news item (most commonly an HTML Webpage).

Performance

- Most of the time is spent downloading the main content of the corpus items.
- A more efficient and faster WebCorpus update.
`result <- corpus.update(yahoonews)`
- The WebCorpus can be continuously updated by setting up a scheduled task/cron job which runs `corpus.update()` in a script.
- For retrieval of more than 10,000 items per feed the *tm.plugin.webmining* package is not suitable and for this purpose, web scraping frameworks like Scrapy (scrapy.org), Heritrix (crawler.archive.org) or Nutch (nutch.apache.org) are much better suited.