

Algorithmic Methods of Data Mining

Web Scrapping

Ioannis Chatzigiannakis

Sapienza University of Rome

Laboratory 5



HyperText Markup Language (HTML)

- ▶ HTML is the most basic building block of the Web.
- ▶ It defines the meaning and structure of web content.
- ▶ Values can contain any type of records.
- ▶ Other technologies besides HTML:
 - ▶ CSS: describe a web page's appearance/presentation.
 - ▶ JavaScript: describe functionality/behavior of a web page.



HyperText

- ▶ "Hypertext" refers to
 - ▶ links that connect web pages to one another.
 - ▶ either within a single website or between websites.
- ▶ Links are a fundamental aspect of the Web.



Markup Language

- ▶ A **markup language** is a computer language that uses tags to define elements within a document.
- ▶ It is human-readable, meaning markup files contain standard words, rather than typical programming syntax.
- ▶ HTML uses "markup" to annotate text, images, and other content for display in a Web browser.

HTML markup includes special "elements"

```
<head>, <title>, <body>, <header>, <footer>, <article>, <section>,
<p>, <div>, <span>, <img>, <aside>, <audio>, <canvas>, <datalist>,
<details>, <embed>, <nav>, <output>, <progress>, <video>, <ul>,
<ol>, <li>
```



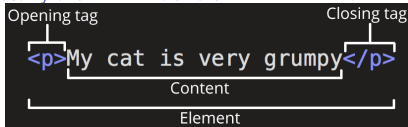
HTML Element

- ▶ An **HTML element** is set off from other text in a document by "tags".
- ▶ Tags consist of the element name surrounded by "<" and ">".
- ▶ The name of an element inside a tag is case insensitive.
- ▶ For example, the <title> tag can be written as <Title>, <TITLE>, or in any other way.

```
<p>My cat is very grumpy</p>
```



Analogy of an HTML element



- ▶ **The opening tag:** The name of the element. Marks where the element begins or starts to take effect.
- ▶ **The content:** The content of the element.
- ▶ **The closing tag:** The same as the opening tag, except that it includes a forward slash before the element name. This marks where the element ends.



Nesting HTML Element

- ▶ Elements can be placed within other elements.
- ▶ This is called **Element nesting**.

```
<p>My cat is <strong>very</strong> grumpy.</p>
```

- ▶ There is a right and wrong way to do nesting.
- ▶ **Tags have to open and close in a way that they are inside or outside one another.**

Wrong usage of Element nesting

```
<p>My cat is <strong>very grumpy.</p></strong>
```



Block versus inline elements

- ▶ Two important categories of HTML elements.
- ▶ Block-level elements form a visible block on a page.
 - ▶ Usually the structural elements on the page.
 - ▶ Appear on a new line.
 - ▶ For example, a block-level element might represent headings, paragraphs, lists, navigation menus, or footers.
- ▶ Inline elements contained within block-level elements.
 - ▶ Surround only small parts of the document's content.
 - ▶ Will not cause a new line to appear in the document.
 - ▶ Typically used with text, for example an <a> element creates a hyperlink, and elements such as or create emphasis.

```
<em>first</em><em>second</em><em>third</em>
```

```
<p>fourth</p><p>fifth</p><p>sixth</p>
```



Empty HTML Elements

- ▶ Empty elements are sometimes called void elements.
- ▶ Not all elements follow the pattern of an opening tag, content, and a closing tag.
- ▶ Some elements consist of a single tag, which is typically used to insert/embed something in the document.
- ▶ For example, the `` element embeds an image file onto a page.

```

```



Attributes of HTML element

- ▶ Elements can also have attributes. Attributes look like this:

Attribute

```
<p class="editor-note">My cat is very grumpy</p>
```

- ▶ Extra information about the element that won't appear in the content.
- ▶ An attribute should have:
 - ▶ A space between it and the element name.
 - ▶ The attribute name, followed by an equal sign.
 - ▶ An attribute value, wrapped with opening and closing quote marks.



The HREF Elements

- ▶ Adding links using the **anchor** `<a>` element.
- ▶ An anchor can make the text it encloses into a hyperlink.
- ▶ Anchors can take a number of attributes, but several are as follows:
 - ▶ **href**: specifies the web address for the link.
 - ▶ **title**: specifies extra information about the link. Appears as a tooltip when a cursor hovers over the element.
 - ▶ **target**: specifies the browsing context used to display the link. If you want to display the linked content in the current tab, just omit this attribute.

```
<a href="https://www.mozilla.org/" title="The Mozilla homepage" target="_blank">The name of the link</a>
```



Boolean Attributes

- ▶ Sometimes you will see attributes written without values. These are called Boolean attributes.
- ▶ Boolean attributes can only have one value, which is generally the same as the attribute name.
- ▶ For example, consider the `disabled` attribute, which you can assign to form input elements.

```
<input type="text" disabled="disabled">
```

```
<input type="text" disabled>
```



Anatomy of an HTML document

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My test page</title>
  </head>
  <body>
    <p>This is my page</p>
  </body>
</html>
```

- ▶ `<html></html>` – wraps all the content on the page. It is sometimes known as the root element.
- ▶ `<head></head>` – page metadata elements.
- ▶ `<body></body>` – all the content.



HTML comments

- ▶ Write comments in the code.
- ▶ Browsers ignore comments, effectively making comments invisible to the user.

```
<p>I'm not inside a comment</p>
```

```
<!-- <p>I am!</p> -->
```



HTML text fundamentals (1)

- ▶ There are six heading elements:
`<h1>`, `<h2>`, `<h3>`, `<h4>`, `<h5>`, `<h6>`.
- ▶ Unordered lists: ``

```
<ul>
  <li>milk</li>
  <li>eggs</li>
  <li>bread</li>
</ul>
```

- ▶ Ordered lists: ``

```
<ol>
  <li>Drive to the end of the road</li>
  <li>Turn right</li>
</ol>
```



HTML text fundamentals (2)

- ▶ Emphasis: ``.
- ▶ Strong: ``
- ▶ Bold: ``
- ▶ Italics: `<i>`
- ▶ Underline: `<u>`



Content Division

- ▶ The HTML Content Division element (`<div>`) – generic container for flow content.
- ▶ Does not inherently represent anything.
- ▶ Used to group content so it can be easily styled using the `class` or `id` attributes.

```
<div class="warning">
  
  <p>Beware of the leopard</p>
</div>
```



Tables

- ▶ The content of every table is enclosed by these two tags : `<table></table>`.
- ▶ The contents of each row is enclosed by these two tags: `<tr></tr>`.
- ▶ The contents of each column, within a row, is enclosed by these two tags: `<td></td>`.

```
<table>
<tr>
  <td>Hi, I'm your first cell.</td>
  <td>I'm your second cell.</td>
  <td>I'm your third cell.</td>
  <td>I'm your fourth cell.</td>
</tr>
</table>
```



Beautiful Soup

- ▶ Beautiful Soup is a Python library for getting data out of HTML, XML, and other markup languages.

```
pip3 install beautifulsoup4
```

- ▶ Additionally, you will need to install a "parser" for interpreting the HTML. To do so, run in the terminal.

```
pip3 install lxml
```



Extracting names and links from an HTML page

```
<table border="1" cellspacing="2" cellpadding="3">
<tbody>
<tr>
<th>Member Name</th>
<th>Birth-Death</th>
</tr>
<tr>
<td><a href="http://bioguide.congress.gov/scripts/biodisplay.pl?index=A000035">ADAMS, George Madison</a></td>
<td>1837-1920</td>
</tr>
<tr>
<td><a href="http://bioguide.congress.gov/scripts/biodisplay.pl?index=A000074">ALBERT, William Julian</a></td>
<td>1816-1879</td>
</tr>
</tbody>
</table>
```



Extracting names and links from an HTML page

```
from bs4 import BeautifulSoup

soup = BeautifulSoup (open("43rd-congress.html"), features="lxml")

links = soup.find_all('a')
for link in links:
    names = link.contents[0]
    fullLink = link.get('href')

    print([names,fullLink])

['ADAMS, George Madison', 'http://bioguide.congress.gov/scripts/biodisplay.pl?index=A000035']
['ALBERT, William Julian', 'http://bioguide.congress.gov/scripts/biodisplay.pl?index=A000074']
```



Retrieving Content

- ▶ We will use the `requests` library.

```
cnt = requests.get("https://www.goodreads.com/book/show/32767.At_the_Mountains_of_Madness")
```

- ▶ Use the contents of the page retrieved

```
soup = BeautifulSoup(cnt.content, features="lxml")
```

- ▶ "Beautifuly" the text:

```
print(soup.prettyfy())
```

- ▶ Write to file:

```
f = open("content.html", "w")
f.write(soup.prettyfy())
f.close()
```



HTTP Agent

- ▶ When we are issuing a HTTP request, the remote server can request information regarding our browser – the HTTP Agent
- ▶ We can specify the User-Agent attribute of an HTTP request.
- ▶ The User-Agent request header contains a characteristic string that allows the network protocol peers to identify the application type, operating system, software vendor or software version of the requesting software user agent.

```
headers = {'User-Agent': 'Mozilla/5.0 (Linux; Android 5.1.1; SM-G928X Build/LMY47X) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/47.0.2526.83 Mobile Safari/537.36'}
countries_response = requests.get("https://www.goodreads.com/book/show/32767.At_the_Mountains_of_Madness", headers=headers)
```



Selenium

- ▶ Selenium is a portable framework for testing web applications.
- ▶ Selenium provides a playback tool for authoring functional tests without the need to learn a test scripting language (Selenium IDE).

```
pip3 install selenium
```

- ▶ A simple usage:

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from webdriver_manager.firefox import GeckoDriverManager

driver = webdriver.Firefox(executable_path=GeckoDriverManager().install())
driver.get("https://www.goodreads.com/book/show/32767.At_the_Mountains_of_Madness")
driver.page_source
```

