Chapter 3 Creating Corpus | Corpus Linguistics

Linguistic data are important to us linguists. Data usually tell us something we don't know, or something we are not sure of. In this chapter, I would like to show you a quick way to extract linguistic data from web pages, which is by now undoubtedly the largest source of textual data available.

While there are many existing text data collections (cf. Structured Corpus and XML), chances are that sometimes you still need to collect your own data for a particular research question. But please note that when you are creating your own corpus for specific research questions, always pay attention to the three important criteria: **representativeness**, **authenticity**, and **size**.

Following the spirit of tidy , we will mainly do our tasks with the libraries of `tidyverse` and `rvests`.

If you are new to **tidyverse** R, please check its official webpage for learning resources.

```
## Uncomment the following line for installation
# install.packages(c("tidyverse", "rvest"))
library(tidyverse)
library(rvest)
```

3.1 HTML Structure

The **H**yper**T**ext **M**arkup **L**anguage, or HTML is the standard markup language for documents designed to be displayed in a web browser.

3.1.1 HTML Syntax

To illustrate the structure of the HTML, please download the sample `html` file from: `demo_data/data-sample-html.html` and first open it with your browser.

```
<!DOCTYPE html>
<html>
  <head>
```

```
  <title>My First HTML </title>

  </head>

  <body>
    <h1> Introduction </h1>
    <p> Have you ever read the source code of a html
page? This is how to get back to the course page: <a
href="https://alvinntnu.github.io/NTNU_ENC2036_LECTURE
S/", target="_blank">ENC2036</a>. </p>
    <h1> Contents of the Page </h1>
    <p> Anything you can say about the page.....</p>
  </body>
</html>
```

An HTML document includes several important elements (cf. Figure 3.1):

- DTD: document type definition which informs the browser about the version of the HTML standard that the document adheres to (e.g., `<!DOCTYPE HTML>`)
- element: the combination of start tag, content, and end tag (e.g, `<title>My First HTML</title>`)
- tag: named braces that enclose the content and define its structural function (e.g., `title`, `body`, `p`)
- attribute: specific properties of the tag, which are often placed in the start end of the element (e.g., `<a href= "index.html"> Homepage </a>`). They are expressed as `name = "value"` pairs.

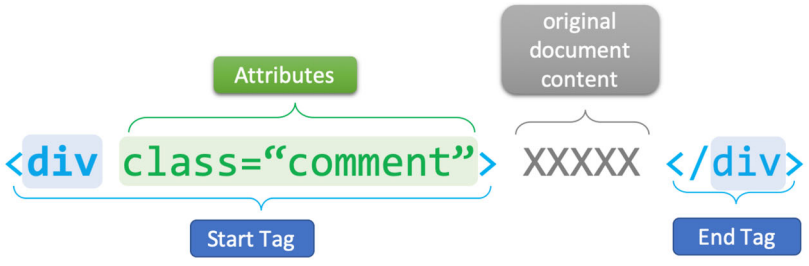


Figure 3.1: Syntax of An HTML Tag Element

An HTML document starts with the root element `<html>`, which splits into two branches, `<head>` and `<body>`.

- Most of the webpage **textual contents** would go into the `<body>` part.
- Most of the web-related codes and metadata (e.g., javascripts, CSS) are often included in the `<head>` part.

All elements need to be strictly nested within each other in a well-formed and valid HTML file, as shown in Figure 3.2.

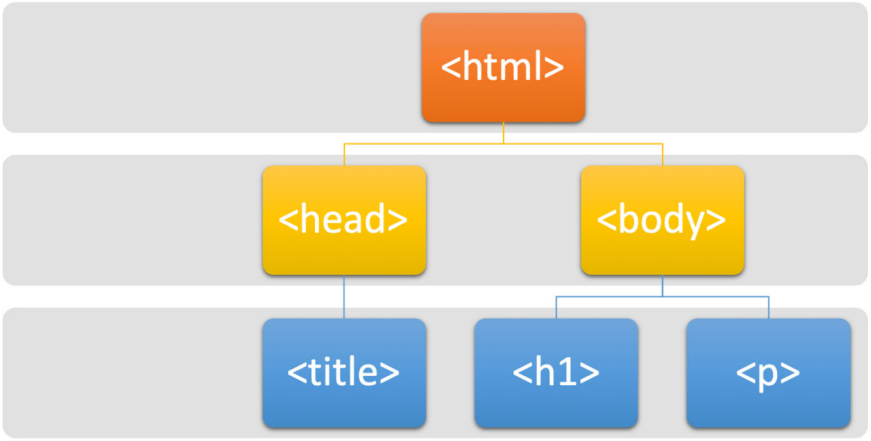


Figure 3.2: Tree Structure of An HTML Document

3.1.2 Tags and Attributes

HTML has plenty of legal tags and attributes. On W3CSchools, there is a complete list of HTML tags and attributes for your reference. Common tags may include:

- Anchor tag `<a>`
- Metadata tag `<meta>`
- External tag `<link>`
- Emphasizing tags `<b>`, `<i>`, `<strong>`
- Paragraph tags `<p>`
- Heading tags `<h1>`, `<h2>`, `<h3>`
- Listing content tags `<ul>`, `<ol>`
- Block tags `<div>`, `<span>`
- Form-related tag `<form>`, `<input>`
- Foreign script tag `<script>`
- Table tag `<table>`, `<th>`, `<tr>`, `<td>`

You probably don't have to know all the detail about the HTML tags and elements. However, in order to scrape the textual data from the Internet, you need to know at least from which parts of HTML elements you need your textual data from on the web pages.

Usually, before you scrape the data from the webpage, bear the following questions in mind:

- From which HTML **elements/tags** would you like to extract the data for corpus construction?
- Do you need the **textual** content of the HTML element?
- Do you need a specific **attribute** of the HTML element?

3.1.3 CSS

Cascading Style Sheet (CSS) is a language for describing the layout of HTML and other markup documents (e.g., XML).

HTML + CSS is by now the standard way to create and design web pages. The idea is that CSS specifies the formats/styles of the HTML elements. The following is an example of the CSS:

```
div.warnings {
  color: pink;
  font-family: "Arial"
  font-size: 120%
}

h1 {
  padding-top: 20px
  padding-bottom: 20px
}
```

You probably would wonder how to link a set of CSS style definitions to an HTML document. There are in general three ways: inline, internal and external. You can learn more about this in W3School.com.

Here I will show you an example of the **internal** method. Below is a CSS style definition for `<h1>`.

```
h1 {
  color: red;
  margin-bottom: 2em;
}
```

We can embed this within a `<style>...</style>` element. Then you put the entire `<style>` element under `<head>` of the HTML file you would like to style.

```
<style>
h1 {
  color: red;
  margin-bottom: 1.5em;
}
</style>
```

After you include the `<style>` in the HTML file, refresh the web page to see if the CSS style works.

- HTML: the language for building web pages
- CSS: the language for styling web pages
- JavaScript: the language for programming web pages

3.2 Web Crawling

In the following demonstration, the text data scraped from the PTT forum is presented as it is without adjustment. However, please note that the language on PTT may strike some readers as profane, vulgar or even offensive.

```
library(tidyverse)
library(rvest)
```

In this tutorial, let's assume that we like to scrape texts from PTT Forum. In particular, we will demonstrate how to scrape texts from the `Gossiping` board of PTT.

```
ptt.url <- "https://www.ptt.cc/bbs/Gossiping"
```

If you use your browser to view PTT Gossiping page, you would see that you need to go through the age verification before you can enter the content page. So, our first job is to pass through this age verification.

- First, we create an `session()` (like we open a browser linking to the page)

```
gossiping.session <- session(ptt.url)
```

- Second, we extract the age verification form from the current page (`form` is also a defined HTML element)

```
gossiping.form <- gossiping.session %>%
  html_node("form") %>%
  html_form
```

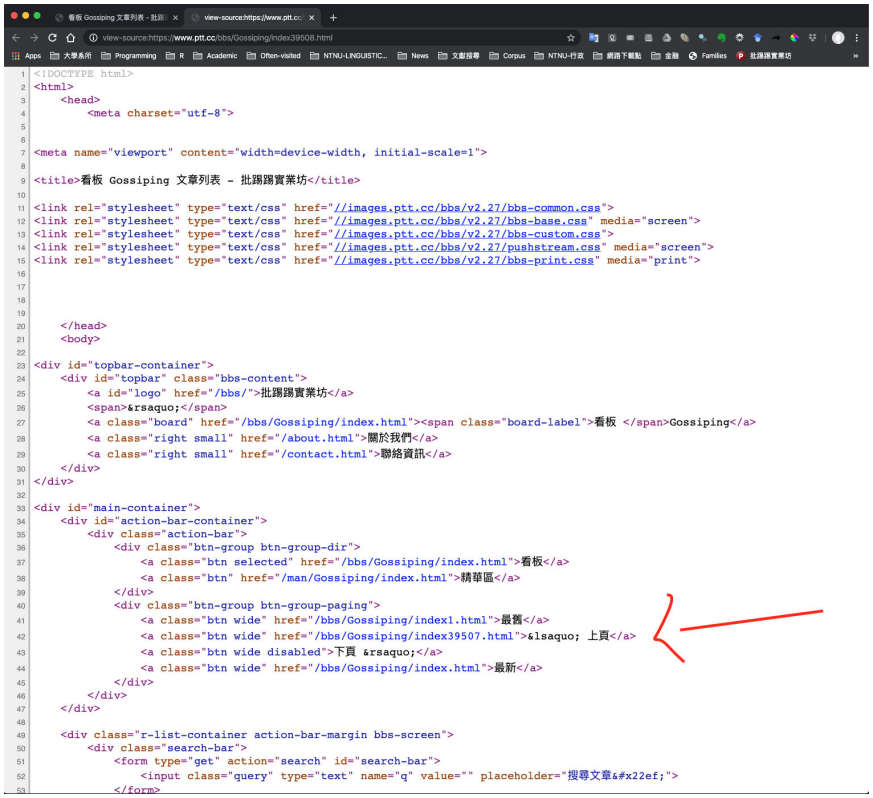
- Then we automatically submit an `yes` to the age verification form in the earlier created `session()` and create another session.

```
gossiping <- session_submit(
  x = gossiping.session,
  form = gossiping.form,
  submit = "yes"
```

```
)

gossiping

<session> https://www.ptt.cc/bbs/Gossiping/index.html
  Status: 200
  Type:   text/html; charset=utf-8
  Size:   17922
```

Now our html `session`, i.e., `gossiping`, should be on the front page of the `Gossiping` board.

Most browsers come with the functionality to inspect the page source (i.e., HTML). This is very useful for web crawling. Before we scrape data from the webpage, we often need to inspect the structure of the web page first. Most importantly, we need to know (a) which HTML elements, or (b) which particular attributes/values of the HTML elements we are interested in .



```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">


<meta name="viewport" content="width=device-width, initial-scale=1">

<title>看板 Gossiping 文章列表 - 批踢踢實業坊</title>

<link rel="stylesheet" type="text/css" href="//images.ptt.cc/bbs/v2.27/bbs-common.css">
<link rel="stylesheet" type="text/css" href="//images.ptt.cc/bbs/v2.27/bbs-base.css" media="screen">
<link rel="stylesheet" type="text/css" href="//images.ptt.cc/bbs/v2.27/bbs-custom.css">
<link rel="stylesheet" type="text/css" href="//images.ptt.cc/bbs/v2.27/pushstream.css" media="screen">
<link rel="stylesheet" type="text/css" href="//images.ptt.cc/bbs/v2.27/bbs-print.css" media="print">



    </head>
    <body>

<div id="topbar-container">
    <div id="topbar" class="bbs-content">
        <a id="logo" href="/bbs/">批踢踢實業坊</a>
        <span>&rsaquo;</span>
        <a class="board" href="/bbs/Gossiping/index.html"><span class="board-label">看板 </span>Gossiping</a>
        <a class="right small" href="/about.html">關於我們</a>
        <a class="right small" href="/contact.html">聯絡資訊</a>
    </div>
</div>

<div id="main-container">
    <div id="action-bar-container">
        <div class="action-bar">
            <div class="btn-group btn-group-dir">
                <a class="btn selected" href="/bbs/Gossiping/index.html">看板</a>
                <a class="btn" href="/man/Gossiping/index.html">精華區</a>
            </div>
            <div class="btn-group btn-group-paging">
                <a class="btn wide" href="/bbs/Gossiping/index1.html">最舊</a>
                <a class="btn wide" href="/bbs/Gossiping/index39507.html">&lsaquo; 上頁</a>
                <a class="btn wide disabled">下頁 &rsaquo;</a>
                <a class="btn wide" href="/bbs/Gossiping/index.html">最新</a>
            </div>
        </div>
    </div>

    <div class="r-list-container action-bar-margin bbs-screen">
        <div class="search-bar">
            <form type="get" action="search" id="search-bar">
                <input class="query" type="text" name="q" value="" placeholder="搜尋文章&#x22ef;">
            </form>
```

- Next we need to find the most recent index page of the board

```
# Decide the number of index pages ----
page.latest <- gossiping %>%
  html_nodes("a") %>% # extract all <a> elements
  html_attr("href") %>%  # extract the attributes
`href`
  str_subset("index[0-9]{2,}\\.html") %>% # find the
`href` with the index number
  str_extract("[0-9]+") %>% # extract the number
  as.numeric()

page.latest
```

```
[1] 39276
```

- On the most recent index page, we need to extract the hyperlinks to the articles

```
# Retreive links -----
link <- str_c(ptt.url, "/index", page.latest, ".html")
links.article <- gossiping %>%
  session_jump_to(link) %>% # move session to the most
recent page
  html_nodes("a") %>% # extract article <a>
  html_attr("href") %>% # extract article <a> `href`
attributes
```

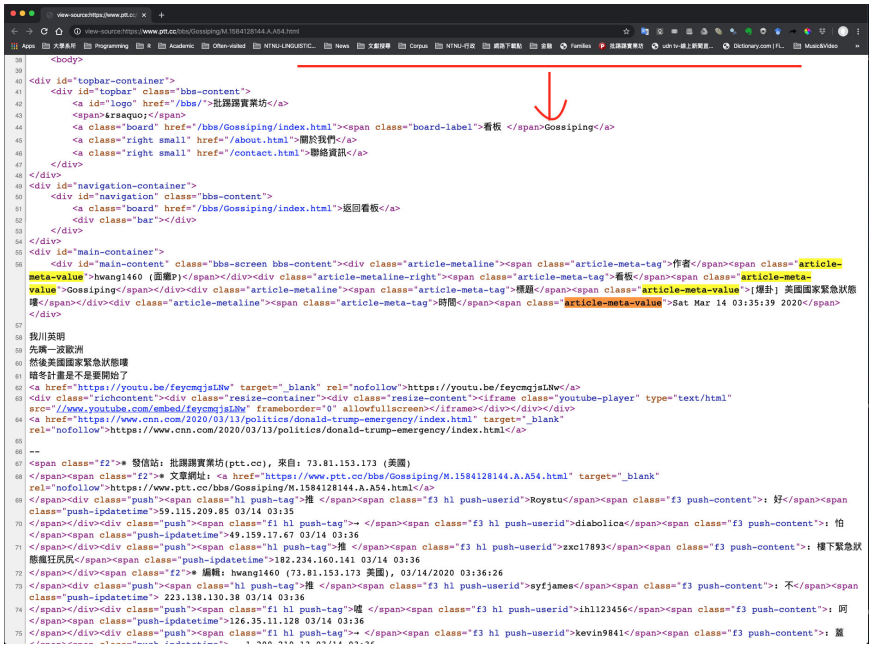
```
  str_subset("[A-z]\\.[0-9]+\\.[A-z]\\.[A-z0-
9]+\\.html") %>% # extract links
  str_c("https://www.ptt.cc",.)
links.article


 [1]
"https://www.ptt.cc/bbs/Gossiping/M.1653590496.A.6BE.h
tml"
 [2]
"https://www.ptt.cc/bbs/Gossiping/M.1653590555.A.F6C.h
tml"
 [3]
"https://www.ptt.cc/bbs/Gossiping/M.1653590986.A.A7D.h
tml"
 [4]
"https://www.ptt.cc/bbs/Gossiping/M.1653591020.A.628.h
tml"
 [5]
"https://www.ptt.cc/bbs/Gossiping/M.1653591084.A.F0D.h
tml"
 [6]
"https://www.ptt.cc/bbs/Gossiping/M.1653591259.A.63B.h
tml"
 [7]
"https://www.ptt.cc/bbs/Gossiping/M.1653591408.A.606.h
tml"
 [8]
"https://www.ptt.cc/bbs/Gossiping/M.1653591758.A.D3E.h
tml"
 [9]
"https://www.ptt.cc/bbs/Gossiping/M.1653591804.A.C0B.h
tml"
[10]
"https://www.ptt.cc/bbs/Gossiping/M.1653592084.A.0A6.h
tml"
[11]
"https://www.ptt.cc/bbs/Gossiping/M.1653592117.A.286.h
tml"
[12]
"https://www.ptt.cc/bbs/Gossiping/M.1653592226.A.795.h
tml"
[13]
"https://www.ptt.cc/bbs/Gossiping/M.1653592247.A.0CD.h
tml"
[14]
```

```
"https://www.ptt.cc/bbs/Gossiping/M.1653592269.A.7A1.h
tml"
[15]
"https://www.ptt.cc/bbs/Gossiping/M.1653592286.A.CCD.h
tml"
[16]
"https://www.ptt.cc/bbs/Gossiping/M.1653592318.A.FD0.h
tml"
[17]
"https://www.ptt.cc/bbs/Gossiping/M.1653592447.A.9AB.h
tml"
[18]
"https://www.ptt.cc/bbs/Gossiping/M.1653592693.A.31E.h
tml"
[19]
"https://www.ptt.cc/bbs/Gossiping/M.1653593324.A.CF2.h
tml"
[20]
"https://www.ptt.cc/bbs/Gossiping/M.1653593596.A.6AE.h
tml"
```

- Next step is to scrape texts from each article hyperlink. Let's consider one link first.

```
article.url <- links.article[1]
temp.html <- gossiping %>%
  session_jump_to(article.url) # link to the article
```

- Now the `temp.html` is like a browser on the article page. Because we are interested in the metadata and the contents of each article, now the question is: where are they in the HTML? We need to go back to the source page of the article HTML again:



- After a closer inspection of the article HTML, we know that:
  - The metadata of the article are included in `<span>` tag elements, belonging to the class `class="article-meta-value"`
  - The contents of the article are included in the `<div>` element, whose ID is `ID="main-content"`
- Now we are ready to extract the metadata of the article.

```
# Extract article metadata
article.header <- temp.html %>%
  html_nodes("span.article-meta-value") %>% # get
<span> of a particular class
  html_text()
article.header
```

```
[1] "jomon817 (Jomon)"                    "Gossiping"
[3] "[問卦] 沒有畢業旅行會造成什麼影響？" "Fri May 27
02:41:34 2022"
```

The metadata of each PTT article in fact includes four pieces of information: author, board name, title, post time. The above code retrieves directly the values of these metadata.

We can retrieve the tags of these metadata values as well:

```
temp.html %>%
  html_nodes("span.article-meta-tag") %>% # get <span>
of a particular class
  html_text()
```

```
[1] "作者" "看板" "標題" "時間"
```

- From the `article.header`, we are able to extract the `author`, `title`, and `time stamp` of the article.

```
article.author <- article.header[1] %>%
str_extract("^[A-z0-9_]+") # athuor
article.title <- article.header[3] # title
article.datetime <- article.header[4] # time stamp

article.author
```

```
[1] "jomon817"
```

```
article.title
```

```
[1] "[問卦] 沒有畢業旅行會造成什麼影響？"
```

```
article.datetime
```

```
[1] "Fri May 27 02:41:34 2022"
```

- Now we extract the main contents of the article

```
article.content <- temp.html %>%
        html_nodes( # article body
          xpath = '//div[@id="main-content"]/node()
[not(self::div|self::span[@class="f2"])]'
        ) %>%
        html_text(trim = TRUE) %>% # extract texts
        str_c(collapse = "") # combine all lines into
one
article.content
```

```
[1] "小魯心想這幾年疫情，根本沒辦法畢旅\n\n小至國小 大到大學 都沒辦法旅台或出國。\n\n一堆有理想的領隊哥哥姐姐們都沒辦法帶團了\n\n沒辦法在ig發一些 為了夢想 之類的話\n\n以後長大沒有這些回憶484很可憐？\n\n--畢旅就是要拿冰火偷偷去女生房間聊天啊"
```

`XPath` (or XML Path Language) is a query language which is useful for addressing and extracting particular elements from XML/HTML

documents. XPath allows you to exploit more features of the hierarchical tree that an HTML file represents in locating the relevant HTML elements. For more information, please see Munzert et al. (2014), Chapter 4.

In the above example, the `XPath` identifies the nodes *under* `<div id = “main-content”>`, but excludes sister nodes that are `<div>` or `<span class=“f2”>`.

These children `<div>` or `<span class=“f2”>` of the `<div id = “main-content”>` include the push comments (推文) of the article, which are not the main content of the article.

- Now we can combine all information related to the article into a data frame

```
article.table <- tibble(
      datetime = article.datetime,
      title = article.title,
      author = article.author,
      content = article.content,
      url = article.url
    )

article.table
```

datetime <chr>	title <chr>	author <chr>
Fri May 27 02:41:34 2022	[問卦] 沒有畢業旅行會造成什麼影響?	jomon817

1 row | 1-3 of 5 columns

- Next we extract the push comments at the end of the article

```
article.push <- temp.html %>%
        html_nodes(xpath = "//div[@class = 'push']")

article.push
```

```
{xml_nodeset (18)}
 [1] <div class="push">\n<span class="f1 hl push-
tag">→ </span><span class="f ...
 [2] <div class="push">\n<span class="f1 hl push-
tag">→ </span><span class="f ...
```

```
 [3] <div class="push">\n<span class="f1 hl push-
tag">→ </span><span class="f ...
 [4] <div class="push">\n<span class="hl push-tag">推
</span><span class="f3 h ...
 [5] <div class="push">\n<span class="f1 hl push-
tag">→ </span><span class="f ...
 [6] <div class="push">\n<span class="hl push-tag">推
</span><span class="f3 h ...
 [7] <div class="push">\n<span class="f1 hl push-
tag">→ </span><span class="f ...
 [8] <div class="push">\n<span class="hl push-tag">推
</span><span class="f3 h ...
 [9] <div class="push">\n<span class="hl push-tag">推
</span><span class="f3 h ...
[10] <div class="push">\n<span class="f1 hl push-
tag">→ </span><span class="f ...
[11] <div class="push">\n<span class="hl push-tag">推
</span><span class="f3 h ...
[12] <div class="push">\n<span class="f1 hl push-
tag">→ </span><span class="f ...
[13] <div class="push">\n<span class="hl push-tag">推
</span><span class="f3 h ...
[14] <div class="push">\n<span class="f1 hl push-
tag">→ </span><span class="f ...
[15] <div class="push">\n<span class="f1 hl push-
tag">→ </span><span class="f ...
[16] <div class="push">\n<span class="f1 hl push-
tag">→ </span><span class="f ...
[17] <div class="push">\n<span class="f1 hl push-
tag">→ </span><span class="f ...
[18] <div class="push">\n<span class="f1 hl push-
tag">→ </span><span class="f ...
```

- We then extract relevant information from each push nodes `article.push`.
  - push types
  - push authors
  - push contents

```
# push tags
push.table.tag <- article.push %>%
  html_nodes("span.push-tag") %>%
  html_text(trim = TRUE) # push types (like or
```

```
dislike)
push.table.tag
```

```
 [1] "→"  "→"  "→"  "推" "→"  "推" "→"  "推" "推" "→"
"推" "→"  "推" "→"  "→"
[16] "→"  "→"  "→"
```

```
# push authors
push.table.author <- article.push %>%
  html_nodes("span.push-userid") %>%
  html_text(trim = TRUE) # author
push.table.author
```

```
 [1] "L1ON"       "ilovemiao"  "Dinenger"   "estupid"
"estupid"
 [6] "Dinenger"   "Dinenger"   "estupid"
"wilson3435" "deepdish"
[11] "a85201207"  "estupid"    "a94037501"  "dearhsu"
"yzelly"
[16] "dbdudsorj"  "dbdudsorj"  "Hans2479"
```

```
# push contents
push.table.content <- article.push %>%
  html_nodes("span.push-content") %>%
  html_text(trim = TRUE)

push.table.content
```

```
 [1] ": 沒差 我也沒去畢業旅行過"
 [2] ": 我只有小學畢旅過 嗚嗚嗚嗚"
 [3] ": 我那年SARS也沒有畢業旅行"
 [4] ": 高中也沒畢旅 那年突然改高二就畢旅"
 [5] ": 當屆高三就直接沒有 白癡學校"
 [6] ": 高三就好好準備考試 玩三小"
 [7] ": 當然是高二出去玩就好"
 [8] ": 爛學校少那三天是能多幾分"
 [9] ": 畢旅不是都高二嗎"
[10] ": 肥宅邊緣人沒差那幾次畢業旅行啦"
[11] ": 我高二欸"
[12] ": 16年前的事了 高二露營 高三畢旅"
[13] ": 生育率降低"
[14] ": 浪費錢 不知道去幹嘛的"
[15] ": 學校不辦不會自己找朋友辦喔? "
```

```
[16] ": 反正沒過幾年就完全忘光 沒去也就算了"
[17] ":  我真的想不起來畢旅..."
[18] ": 根本沒差"
```

```
# push time
push.table.datetime <- article.push %>%
  html_nodes("span.push-ipdatetime") %>%
  html_text(trim = TRUE) # push time stamp
push.table.datetime
```

```
 [1] "114.136.40.219 05/27 02:42"  "61.227.217.105
05/27 02:42"
 [3] "27.240.216.189 05/27 02:42"  "220.135.54.181
05/27 02:46"
 [5] "220.135.54.181 05/27 02:47"  "27.240.216.189
05/27 02:47"
 [7] "27.240.216.189 05/27 02:48"  "220.135.54.181
05/27 02:50"
 [9] "150.117.212.52 05/27 02:50"  "220.134.89.190
05/27 02:50"
[11] "49.159.0.221 05/27 02:50"    "220.135.54.181
05/27 02:51"
[13] "42.74.224.19 05/27 02:52"    "180.218.206.44
05/27 02:52"
[15] "49.217.73.101 05/27 03:00"   "111.249.161.81
05/27 03:08"
[17] "111.249.161.81 05/27 03:08"  "118.150.177.245
05/27 04:19"
```

- Finally, we combine all into one Push data frame.

```
push.table <- tibble(
      tag = push.table.tag,
      author = push.table.author,
      content = push.table.content,
      datetime = push.table.datetime,
      url = article.url)
```

```
push.table
```

tag	author	content
<chr>	<chr>	<chr>
→	L1ON	: 沒差 我也沒去畢業旅行過

tag <chr>	author <chr>	content <chr>
→	ilovemiao	: 我只有小學畢旅過 嗚嗚嗚嗚
→	Dinenger	: 我那年SARS也沒有畢業旅行
推	estupid	: 高中也沒畢旅 那年突然改高二就畢旅
→	estupid	: 當屆高三就直接沒有 白癡學校
推	Dinenger	: 高三就好好準備考試 玩三小
→	Dinenger	: 當然是高二出去玩就好
推	estupid	: 爛學校少那三天是能多幾分
推	wilson3435	: 畢旅不是都高二嗎
→	deepdish	: 肥宅邊緣人沒差那幾次畢業旅行啦

Next
12
Previous
1-10 of 18 rows | 1-3 of 5 columns

3.3 Functional Programming

It should now be clear that there are several routines that we need to do again and again if we want to collect text data in large amounts:

- For each index page, we need to extract all the article hyperlinks of the page.
- For each article hyperlink, we need to extract the article content, metadata, and the push comments.

So, it would be great if we can wrap these two routines into two functions.

3.3.1 `extract_art_links()`

- `extract_art_links()`: This function takes an HTML session `session` and an index page of the PTT Gossiping `index_page` as the arguments and extract all article links from the index page. It returns a vector of article links.

```
extract_art_links <- function(index_page, session){
  links.article <- session %>%
```

```
  session_jump_to(index_page) %>%
  html_nodes("a") %>%
  html_attr("href") %>%
  str_subset("[A-z]\\.[0-9]+\\.[A-z]\\.[A-z0-
9]+\\.html") %>%
  str_c("https://www.ptt.cc",.)

  return(links.article)
}
```

For example, we can extract all the article links from the most recent index page:

```
# Get index page
cur_index_page <- str_c(ptt.url, "/index",
page.latest, ".html")
# Get all article links from the most recent index
page
cur_art_links <-extract_art_links(cur_index_page,
gossiping)
cur_art_links
```

```
 [1]
"https://www.ptt.cc/bbs/Gossiping/M.1653590496.A.6BE.h
tml"
 [2]
"https://www.ptt.cc/bbs/Gossiping/M.1653590555.A.F6C.h
tml"
 [3]
"https://www.ptt.cc/bbs/Gossiping/M.1653590986.A.A7D.h
tml"
 [4]
"https://www.ptt.cc/bbs/Gossiping/M.1653591020.A.628.h
tml"
 [5]
"https://www.ptt.cc/bbs/Gossiping/M.1653591084.A.F0D.h
tml"
 [6]
"https://www.ptt.cc/bbs/Gossiping/M.1653591259.A.63B.h
tml"
 [7]
"https://www.ptt.cc/bbs/Gossiping/M.1653591408.A.606.h
tml"
 [8]
"https://www.ptt.cc/bbs/Gossiping/M.1653591758.A.D3E.h
```

```
tml"
 [9]
"https://www.ptt.cc/bbs/Gossiping/M.1653591804.A.C0B.h
tml"
[10]
"https://www.ptt.cc/bbs/Gossiping/M.1653592084.A.0A6.h
tml"
[11]
"https://www.ptt.cc/bbs/Gossiping/M.1653592117.A.286.h
tml"
[12]
"https://www.ptt.cc/bbs/Gossiping/M.1653592226.A.795.h
tml"
[13]
"https://www.ptt.cc/bbs/Gossiping/M.1653592247.A.0CD.h
tml"
[14]
"https://www.ptt.cc/bbs/Gossiping/M.1653592269.A.7A1.h
tml"
[15]
"https://www.ptt.cc/bbs/Gossiping/M.1653592286.A.CCD.h
tml"
[16]
"https://www.ptt.cc/bbs/Gossiping/M.1653592318.A.FD0.h
tml"
[17]
"https://www.ptt.cc/bbs/Gossiping/M.1653592447.A.9AB.h
tml"
[18]
"https://www.ptt.cc/bbs/Gossiping/M.1653592693.A.31E.h
tml"
[19]
"https://www.ptt.cc/bbs/Gossiping/M.1653593324.A.CF2.h
tml"
[20]
"https://www.ptt.cc/bbs/Gossiping/M.1653593596.A.6AE.h
tml"
```

3.3.2 `extract_article_push_tables()`

- `extract_article_push_tables()`: This function takes an article link `link` as the argument and extracts the metadata, textual contents, and pushes of the article. It returns a *list* of two elements —article and push data frames.

```
extract_article_push_tables <- function(link){
  article.url <- link
  temp.html <- gossiping %>%
session_jump_to(article.url) # link to the www
  # article header
  article.header <- temp.html %>%
    html_nodes("span.article-meta-value") %>% # meta
info regarding the article
    html_text()

  # article meta
  article.author <- article.header[1] %>%
str_extract("^[A-z0-9_]+") # athuor
  article.title <- article.header[3] # title
  article.datetime <- article.header[4] # time stamp

  # article content
  article.content <- temp.html %>%
    html_nodes( # article body
      xpath = '//div[@id="main-content"]/node()
[not(self::div|self::span[@class="f2"])]'
    ) %>%
    html_text(trim = TRUE) %>%
    str_c(collapse = "")

  # Merge article table
  article.table <- tibble(
        datetime = article.datetime,
        title = article.title,
        author = article.author,
        content = article.content,
        url = article.url
      )

  # push nodes
  article.push <- temp.html %>%
    html_nodes(xpath = "//div[@class = 'push']") #
extracting pushes
    # NOTE: If CSS is used, div.push does a lazy match
(extracting div.push.... also)

  # push tags
  push.table.tag <- article.push %>%
    html_nodes("span.push-tag") %>%
    html_text(trim = TRUE) # push types (like or
```

```
dislike)

  # push author id
  push.table.author <- article.push %>%
    html_nodes("span.push-userid") %>%
    html_text(trim = TRUE) # author

  # push content
  push.table.content <- article.push %>%
    html_nodes("span.push-content") %>%
    html_text(trim = TRUE)

  # push datetime
  push.table.datetime <- article.push %>%
    html_nodes("span.push-ipdatetime") %>%
    html_text(trim = TRUE) # push time stamp

  # merge push table
  push.table <- tibble(
          tag = push.table.tag,
          author = push.table.author,
          content = push.table.content,
          datetime = push.table.datetime,
          url = article.url
        )

  # return

  return(list(article.table = article.table,
              push.table = push.table))
}#endfunc
```

For example, we can get the `article` and `push` tables from the first article link:

```
extract_article_push_tables(cur_art_links[1])
```

```
$article.table
# A tibble: 1 × 5
  datetime                title        author
content            url
  <chr>                   <chr>        <chr>   <chr>
<chr>
1 Fri May 27 02:41:34 2022 [問卦] 沒有… jomon8… "小魯心
想這幾年疫… https://www.…
```

```
$push.table
# A tibble: 18 × 5
   tag   author     content
datetime    url
   <chr> <chr>      <chr>
<chr>       <chr>
 1 →     L1ON       : 沒差 我也沒去畢業旅行過
114.136.40… https://w…
 2 →     ilovemiao  : 我只有小學畢旅過 嗚嗚嗚嗚
61.227.217… https://w…
 3 →     Dinenger   : 我那年SARS也沒有畢業旅行
27.240.216… https://w…
 4 推    estupid    : 高中也沒畢旅 那年突然改高二就畢旅
220.135.54… https://w…
 5 →     estupid    : 當屆高三就直接沒有 白癡學校
220.135.54… https://w…
 6 推    Dinenger   : 高三就好好準備考試 玩三小
27.240.216… https://w…
 7 →     Dinenger   : 當然是高二出去玩就好
27.240.216… https://w…
 8 推    estupid    : 爛學校少那三天是能多幾分
220.135.54… https://w…
 9 推    wilson3435 : 畢旅不是都高二嗎
150.117.21… https://w…
10 →     deepdish   : 肥宅邊緣人沒差那幾次畢業旅行啦
220.134.89… https://w…
11 推    a85201207  : 我高二欸
49.159.0.2… https://w…
12 →     estupid    : 16年前的事了 高二露營 高三畢旅
220.135.54… https://w…
13 推    a94037501  : 生育率降低
42.74.224.… https://w…
14 →     dearhsu    : 浪費錢 不知道去幹嘛的
180.218.20… https://w…
15 →     yzelly     : 學校不辦不會自己找朋友辦喔?
49.217.73.… https://w…
16 →     dbdudsorj  : 反正沒過幾年就完全忘光 沒去也就算了
111.249.16… https://w…
17 →     dbdudsorj  :  我真的想不起來畢旅...
111.249.16… https://w…
18 →     Hans2479   : 根本沒差
118.150.17… https://w…
```

3.3.3 Streamline the Codes

Now we can simplify our codes quite a bit:

```
# Get index page
cur_index_page <- str_c(ptt.url, "/index",
page.latest, ".html")

# Scrape all article.tables and push.tables from each
article hyperlink
cur_index_page %>%
  extract_art_links(session = gossiping) %>%
  map(extract_article_push_tables) -> ptt_data

# number of articles on this index page
length(ptt_data)
```

```
[1] 20
```

```
# Check the first contents of 1st hyperlink
ptt_data[[1]]$article.table
```

datetime <chr>	title <chr>	author <chr>
Fri May 27 02:41:34 2022	[問卦] 沒有畢業旅行會造成什麼影響?	jomon817

1 row | 1-3 of 5 columns

```
ptt_data[[1]]$push.table
```

tag <chr>	author <chr>	content <chr>
→	L1ON	: 沒差 我也沒去畢業旅行過
→	ilovemiao	: 我只有小學畢旅過 嗚嗚嗚嗚
→	Dinenger	: 我那年SARS也沒有畢業旅行
推	estupid	: 高中也沒畢旅 那年突然改高二就畢旅
→	estupid	: 當屆高三就直接沒有 白癡學校
推	Dinenger	: 高三就好好準備考試 玩三小
→	Dinenger	: 當然是高二出去玩就好

tag <chr>	author <chr>	content <chr>
推	estupid	: 爛學校少那三天是能多幾分
推	wilson3435	: 畢旅不是都高二嗎
→	deepdish	: 肥宅邊緣人沒差那幾次畢業旅行啦

Next
12
Previous
1-10 of 18 rows | 1-3 of 5 columns

- Finally, the last thing we can do is to combine all article tables from each index page into one; and all push tables into one for later analysis.

```
# Merge all article.tables into one
article.table.all <- ptt_data %>%
  map(function(x) x$article.table) %>%
  bind_rows

# Merge all push.tables into one
push.table.all <- ptt_data %>%
  map(function(x) x$push.table) %>%
  bind_rows

article.table.all
```

datetime <chr>	title <chr>
Fri May 27 02:41:34 2022	[問卦] 沒有畢業旅行會造成什麼影響?
Fri May 27 02:42:33 2022	Re: [問卦] 清冠臨床數據有12人很足夠了吧?
Fri May 27 02:49:44 2022	Re: [問卦] 沒有畢業旅行會造成什麼影響?
Fri May 27 02:50:18 2022	[問卦] 有人被雨聲吵醒然後睡不著的嗎?
Fri May 27 02:51:22 2022	[問卦] 用疫病取代戰爭進行人口縮減你要嗎?

datetime <chr>	title <chr>
Fri May 27 02:54:15 2022	[問卦] 中英文之間不加空白的人在想什麼
Fri May 27 02:56:46 2022	Re: [問卦] 從郭彥均事件看台灣社會是不是病了
Fri May 27 03:02:36 2022	[問卦] 政大？就這?
Fri May 27 03:03:21 2022	Re: [問卦] 有人被雨聲吵醒然後睡不著的嗎?
Fri May 27 03:08:02 2022	Re: [問卦] 醫護想要的是什麼?

Next
12
Previous
1-10 of 20 rows | 1-2 of 5 columns

```
push.table.all
```

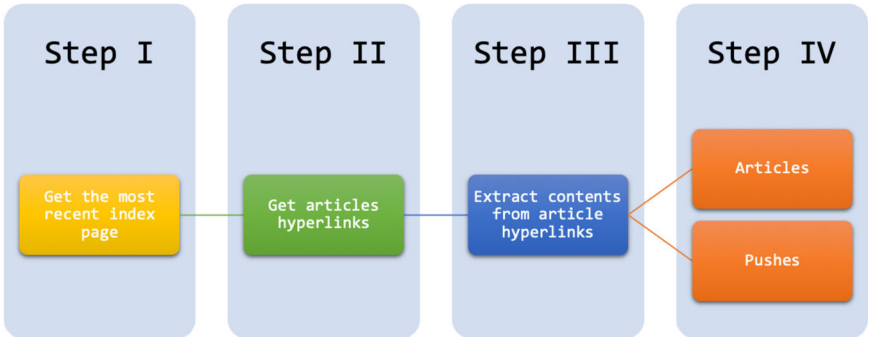
tag <chr>	author <chr>	content <chr>
→	L1ON	: 沒差 我也沒去畢業旅行過
→	ilovemiao	: 我只有小學畢旅過 嗚嗚嗚嗚
→	Dinenger	: 我那年SARS也沒有畢業旅行
推	estupid	: 高中也沒畢旅 那年突然改高二就畢旅
→	estupid	: 當屆高三就直接沒有 白癡學校
推	Dinenger	: 高三就好好準備考試 玩三小
→	Dinenger	: 當然是高二出去玩就好
推	estupid	: 爛學校少那三天是能多幾分
推	wilson3435	: 畢旅不是都高二嗎
→	deepdish	: 肥宅邊緣人沒差那幾次畢業旅行啦

Next
123456

...
31
Previous
1-10 of 301 rows | 1-3 of 5 columns

There is still one problem with the Push data frame. Right now it is still not very clear how we can match the pushes to the articles from which they were extracted. The only shared index is the `url`. It would be better if all the articles in the data frame have their own unique indices and in the Push data frame each push comment corresponds to a particular article index.

The following graph summarizes our work flowchart for PTT Gossipping Scraping:



3.4 Save Corpus

You can easily save your scraped texts in a `CSV` format.

```
# Save ------
write_csv(article.table, path =
"PTT_GOSSIPPING_ARTICLE.csv")
write_csv(push.table, path =
"PTT_GOSSIPPING_PUSH.csv")
```

3.5 Additional Resources

Collecting texts and digitizing them into machine-readable files is only the initial step for corpus construction. There are many other things that need to be considered to ensure the effectiveness and the sustainability of the corpus data. In particular, I would like to point you to a very useful resource, Developing Linguistic Corpora: A Guide to Good Practice, compiled by Martin Wynne. Other important issues in corpus creation include:

1. Adding linguistic **annotations** to the corpus data (cf. Leech's Chapter 2)

2. **Metadata** representation of the documents (cf. Burnard's Chapter 4)
3. Spoken corpora (cf. Thompson's Chapter 5)
4. Technical parts for corpus creation (cf. Sinclair's Appendix)

3.6 Final Remarks

1. Please pay attention to the ethical aspects involved in the process of web crawling (esp. with personal private matters).
2. If the website has their own API built specifically for one to gather data, use it instead of scraping.
3. Always read the terms and conditions provided by the website regarding data gathering.
4. Always be gentle with the data scraping (e.g., off-peak hours, spacing out the requests)
5. Value the data you gather and treat the data with respect.

Exercise 3.1 Can you modify the R codes so that the script can automatically scrape more than one index page?
Exercise 3.2 Please utilize the code from Exercise 3.1 and collect all texts on `PTT/Gossipings` from **3** index pages. Please have the articles saved in `PTT_GOSSIPING_ARTICLE.csv` and the pushes saved in `PTT_GOSSIPING_PUSH.csv` under your working directory.

Also, at the end of your code, please also output in the Console the corpus size, including both the articles and the pushes. Please provide the total number of *characters* of all your PTT text data collected (Note: You DO NOT have to do the word segmentation yet. Please use the characters as the base unit for corpus size.)

Hint: `nchar()`
Your script may look something like:

```
# I define my own `scrapePTT()` function:
# ptt_url: specify the board to scrape texts from
# num_index_page: specify the number of index pages to
be scraped
# return: list(article, push)

PTT_data <-scrapePTT(ptt_url =
"https://www.ptt.cc/bbs/Gossiping", num_index_page =
3)

PTT_data$article %>% head
```

datetime <chr>
Fri May 27 01:37:29 2022
Fri May 27 01:41:27 2022
Fri May 27 01:42:06 2022
Fri May 27 01:43:17 2022
Fri May 27 01:43:21 2022
Fri May 27 01:43:56 2022

6 rows | 1-1 of 5 columns

```
PTT_data$push %>% head
```

tag <chr>	author <chr>	content <chr>
推	Beanoodle	: 愛你
→	Eclipsis	: 今天美國滿多畢典直播的 現在可以yt上
→	Eclipsis	: 搜下diploma ceremony live
推	DPP48	: 席巴女王的進場
推	lovetina	: "港姐"加冕的音樂
推	hosen	: 已經是了不用明日

6 rows | 1-3 of 5 columns

```
# corpus size

PTT_data$article$content %>% nchar %>% sum


[1] 28895
```

Exercise 3.3 Please choose a website (other than PTT) you are interested in and demonstrate how you can use R to retrieve textual data from the site. The final scraped text collection could be from only one static web page. The purpose of this exercise is to show that you know how to parse the HTML structure of the web page and retrieve the data you need from the website.

References

Munzert, S., Rubba, C., Meißner, P., & Nyhuis, D. (2014). *Automated data collection with R: A practical guide to web scraping and text mining*. John Wiley & Sons.