데이터

1. 스크래핑
1.1. Web
1.2. Crawling
1.3. Scrapping

2. 텍스트 분석
2.1. KoNLPy, NLTK
2.2. Tokenization
2.3. 불용어/연어

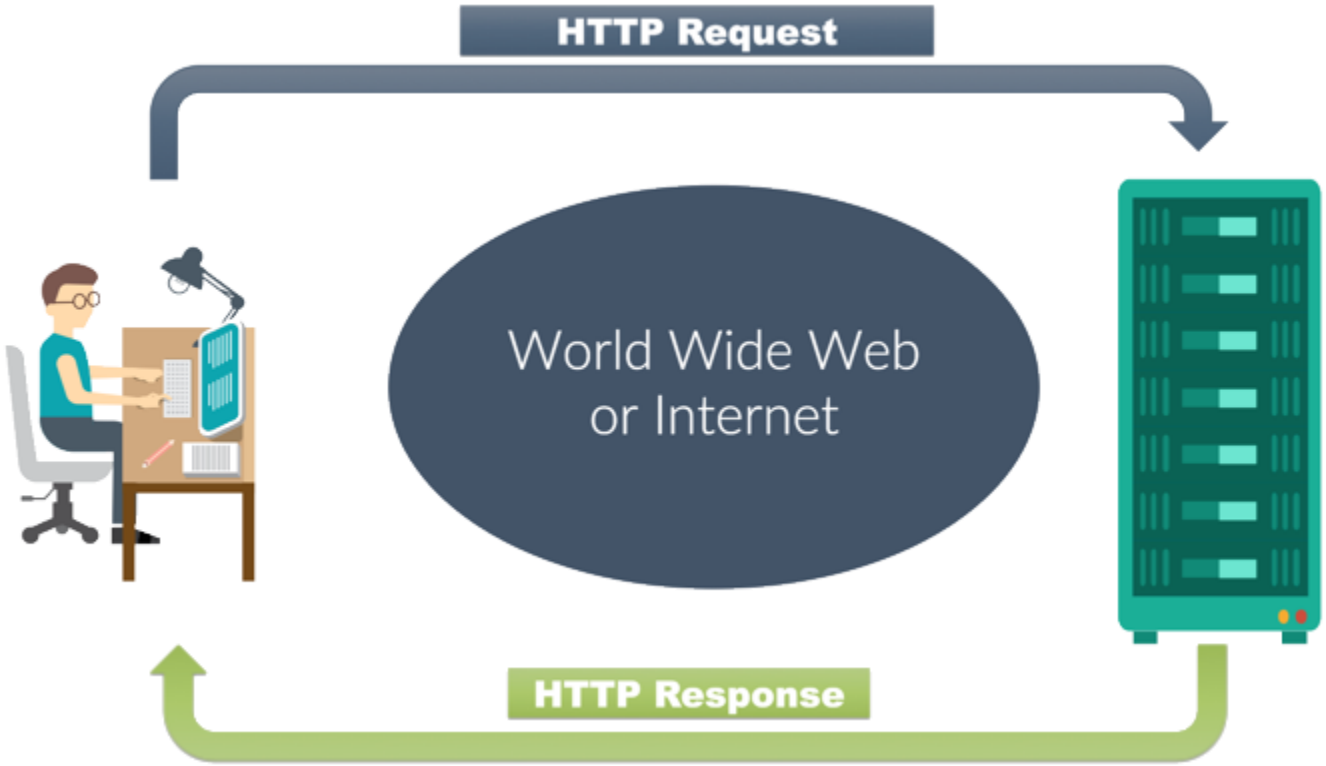
3. 정보검색
3.1. Vector Space Model
3.2. Inverted Document
3.3. Indexing
3.4. Retrieving

1. 스크래핑

1.1. HTTP
1.2. Crawling
1.3. Scrapping

1.1. HTTP

■ HTTP Architecture



■ 예제

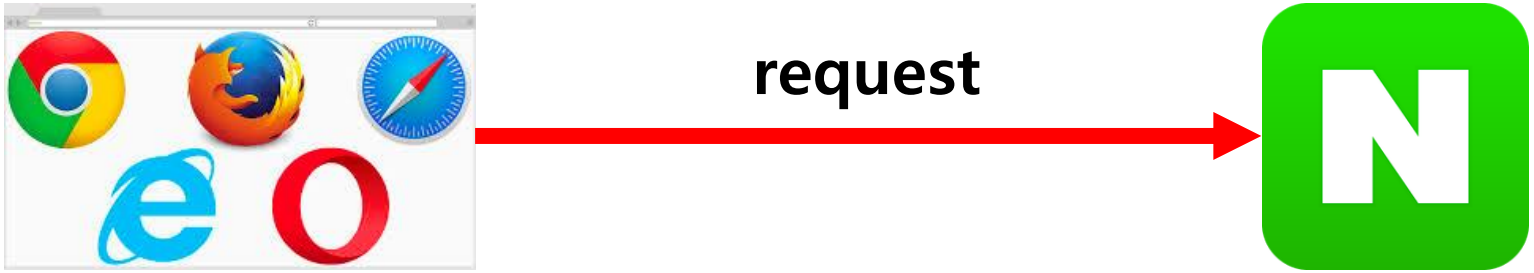
Client

HTTP

http://www.naver.com

Server

○ Request



▼ General

Request URL: https://www.naver.com/

Request Method: GET

Status Code: 200

Remote Address: 23.35.221.113:443

Referrer Policy: origin

▼ Request Headers

:authority: www.naver.com

:method: GET

:path: /

:scheme: https

accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8

accept-encoding: gzip, deflate, br

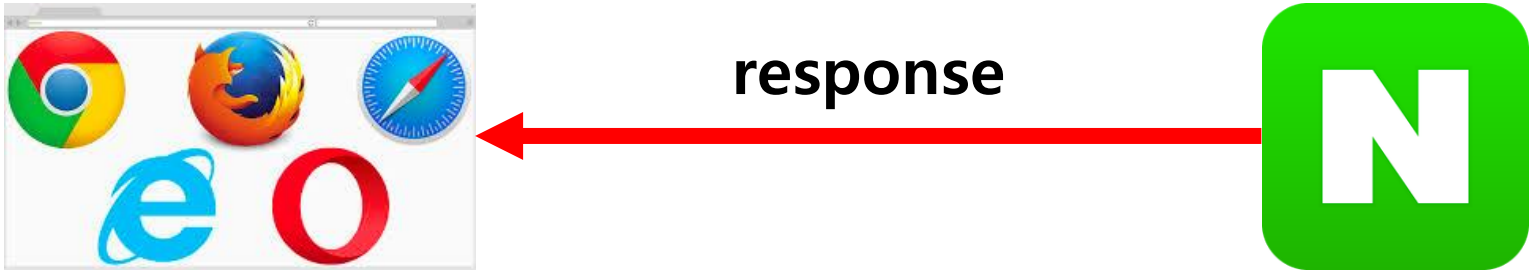
accept-language: ko-KR,ko;q=0.9,en-US;q=0.8,en;q=0.7

cookie: npic=79bi60ql0lJkDEszX/3bqloz0I/RKFahZzkp84I+Dv45TGv/A1T4U11FknSWd764CA==; NNB=AFU2GPITRSAVS; __date_rename_layer=201782; AS
hfaJm5VzyHz7K6/x4+d; PM_CK_rcode=09290610; NID_SES=AAABc24B/kcGVTdYOUOnVCWLtob2wMX39p9CjP3ylWPUbVgBdRgkKE4I9EpmjpIkXjSsSmINGzVImQt9
As7SKEC5lKY2d7Fmia48EwC70L8dp/RUzGZ26tq0gnzYde9y0Jo0kPllozxD+1O8XyDjxeyij2//L9HIpJhVtUEYnAF+ByXoSeGoCc6+6e4ZDquxlzfHvgLM5RHwulLJ/BU
mil/Ao0M32AakjNMPV0nl0PESUwvFQNc/u1QW5VjRHHe6VXMdFWU2aRmcmHivdkwCLlaDB/XKbLYERp4sIDM99mkUwCvsCzfRfys2BJyQ==

upgrade-insecure-requests: 1

user-agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.99 Safari/537.36

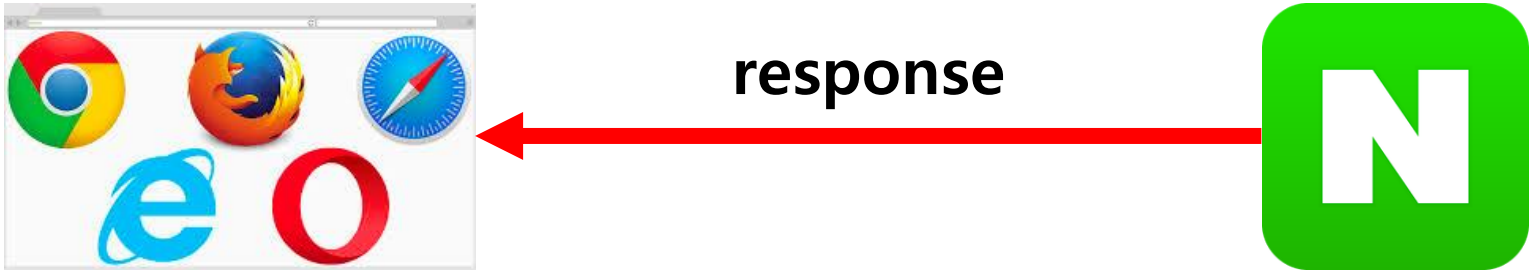
○ Response



▼ Response Headers

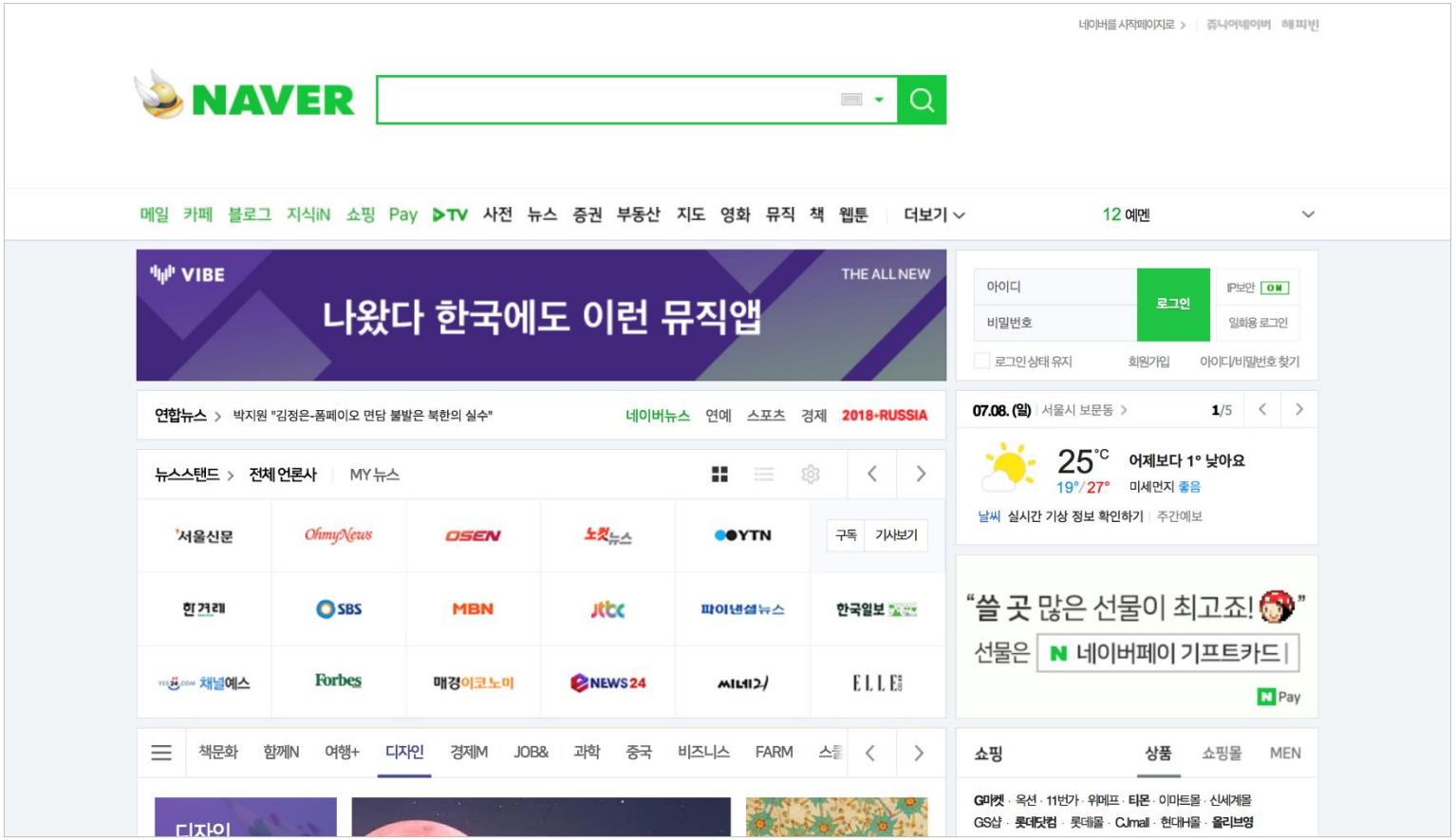
cache-control: no-cache, no-store, must-revalidate
content-encoding: gzip
content-length: 27896
content-type: text/html; charset=UTF-8
date: Sun, 08 Jul 2018 05:36:18 GMT
p3p: CP="CAO DSP CURa ADMa TAIa PSAa OUR LAW STP PHY ONL UNI PUR FIN COM NAV INT DEM STA PRE"
pragma: no-cache
referrer-policy: unsafe-url
server: NWS
status: 200
strict-transport-security: max-age=31536000; preload
vary: Accept-Encoding
x-frame-options: SAMEORIGIN

○ Content



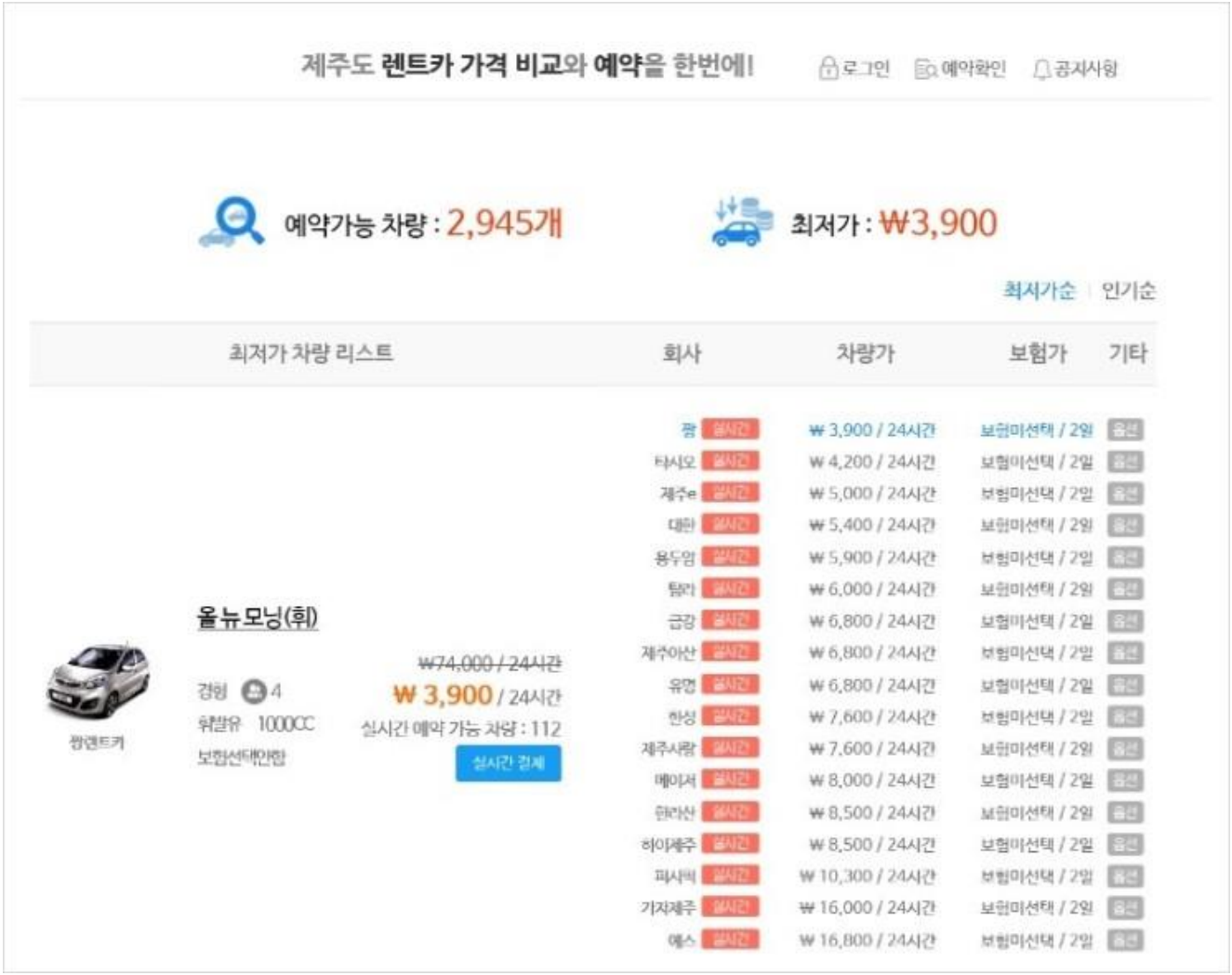

```
<div class="special_bg">
<div class="area_flex">
<div class="area_logo">
<h1>
<a data-clk="top.logo" href="/"><span class="naver_logo">네이버</span></a>
</h1>
</div>
<div class="area_links">
<a data-clk="top.mkhome" href="http://help.naver.com/support/alias/contents2/naverhome/naverhome_1.naver" class="al_favorite">네이버를 시작페이지로<span class="al_ico_link"></span></a>
<span class="al_bar"></span>
<a data-clk="top.jrnaver" href="http://jr.naver.com" class="al_jr"><span class="blind">쥬니어네이버</span><span class="al_ico"></span></a>
<a data-clk="top.happybean" href="http://happybean.naver.com/main/SectionMain.nhn" class="al_happybean"><span class="blind">해피빈</span><span class="al_ico"></span></a>
</div>
<div id="search" class="search">
<!--자동완성 입력창-->

<form id="sform" name="sform" action="https://search.naver.com/search.naver" method="get">

    <fieldset>
        <legend class="blind">검색</legend>
        <select id="where" name="where" title="검색 범위 선택" class="blind">
            <option value="nexearch" selected="selected">통합검색</option><option value="post">블로그</option><option value="cafeblog">카페</option><option value="cafe">- 카페명</option><option
value="article">- 카페글</option><option value="kin">지식iN</option><option value="news">뉴스</option><option value="web">사이트</option><option value="category">- 카테고리</option><option value="site">- 사이트
</option><option value="movie">영화</option><option value="webkr">웹문서</option><option value="dic">사전</option><option value="100">- 백과사전</option><option value="endic">- 영어사전</option><option
value="eedic">- 영영사전</option><option value="krdic">- 국어사전</option><option value="jpdic">- 일본어사전</option><option value="hanja">- 한자사전</option><option value="terms">- 용어사전</option><option
value="book">책</option><option value="music">음악</option><option value="doc">전문자료</option><option value="shop">쇼핑</option><option value="local">지역</option><option value="video">동영상</option><option
value="image">이미지</option><option value="mypc">내PC</option><optgroup label="스마트 파인더"><option value="movie">영화</option><option value="auto">자동차</option><option value="game">게임</option><option
value="health">건강</option><option value="people">인물</option></optgroup><optgroup label="네이버 랩"><option>긍정부정검색</option></optgroup>
        </select>
        <input type="hidden" id="sm" name="sm" value="top_hty" />
        <input type="hidden" id="fbm" name="fbm" value="0" />
        <input type="hidden" id="acr" name="acr" value="" disabled="disabled" />
        <input type="hidden" id="acq" name="acq" value="" disabled="disabled" />
        <input type="hidden" id="qdt" name="qdt" value="" disabled="disabled" />
        <input type="hidden" id="ie" name="ie" value="utf8" />
        <input type="hidden" id="acir" name="acir" value="" disabled="disabled" />
        <input type="hidden" id="os" name="os" value="" disabled="disabled" />
        <input type="hidden" id="bid" name="bid" value="" disabled="disabled" />
        <input type="hidden" id="pkid" name="pkid" value="" disabled="disabled" />
        <input type="hidden" id="eid" name="eid" value="" disabled="disabled" />
        <input type="hidden" id="mra" name="mra" value="" disabled="disabled" />
        <span class="green_window">
            <input id="query" name="query" type="text" title="검색어 입력" maxlength="255" class="input_text" tabindex="1" accesskey="s" style="ime-mode:active;" autocomplete="off"
onclick="document.getElementById('fbm').value=1;" value="" />
        </span>
        <div id="nautocomplete" class="autocomplete">
            <!-- 자동완성 열린 경우 fold 클래스 추가, 딤드인 경우 dim 추가 -->
            <a href="javascript:;" role="button" tabindex="2" class="btn_arw _btn_arw fold"><span class="blind _text">자동완성 펼치기</span><span class="ico_arr"></span></a>
        </div>
        <button id="search_btn" type="submit" title="검색" tabindex="3" class="sch_smit" onmouseover="this.className='sch_smit over'" onmousedown="this.className='sch_smit down'"
onmouseout="this.className='sch_smit'" onclick="clickcr(this,'sch.action','','',event);"><span class="blind">검색</span><span class="ico_search_submit"></span></button>
    </fieldset>
</form>
<!--자동완성 입력창-->
<!--한글입력기 -->
<a href="javascript:;" id="ke_kbd_btn" role="button" class="btn_keyboard" onclick="nx_ime_load(this)" data-clk="sch.ime"><span class="blind">한글 입력기</span><span class="ico_keyboard"></span></a>
<style type="text/css" id="_nx_kbd_style"></style>
<div id="_nx_kbd" style="display:none;"></div>
<!--한글입력기 -->
```

○ Rendering

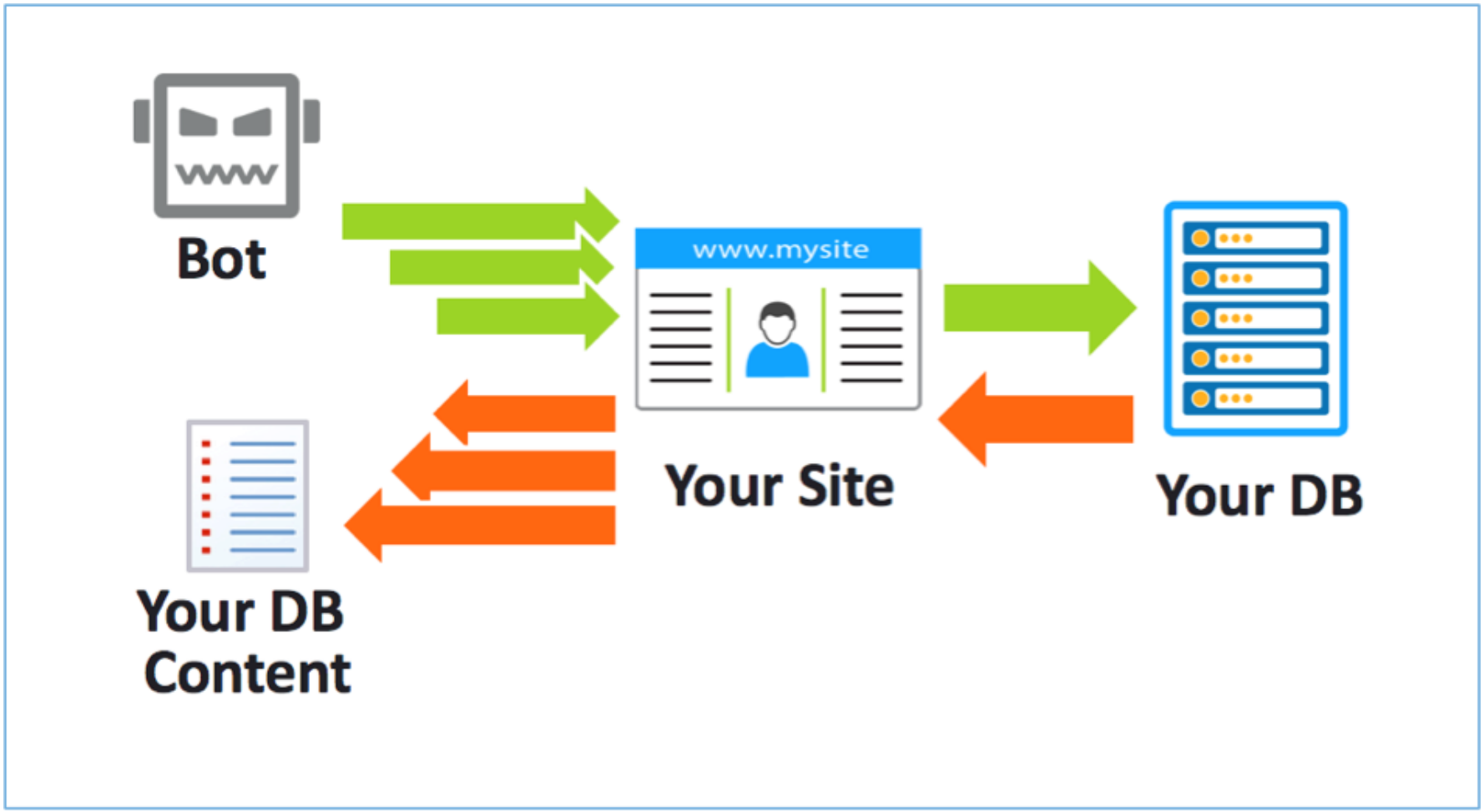

NAVER
메일 카페 블로그 지식iN 쇼핑 Pay TV 사전 뉴스 증권 부동산 지도 영화 뮤직 책 웹툰 더보기
12 예멘
VIBE
THE ALL NEW
나왔다 한국에도 이런 뮤직앱
아이디
비밀번호
로그인
로그인 상태 유지
회원가입
아이디/비밀번호 찾기
연합뉴스 박지원 "김정은-폼페이오 면담 불발은 북한의 실수"
네이버뉴스 연예 스포츠 경제 2018-RUSSIA
뉴스스탠드 전체 언론사 MY 뉴스
서울신문 OhmyNews OSEN 노컷뉴스 YTN
한겨레 SBS MBN jtbc 파이낸셜뉴스 한국일보
채널예스 Forbes 매경이코노미 NEWS24 시네21 ELLE
07.08. (일) 서울시 보문동
25°C
19°/27°
어제보다 1° 낮아요
미세먼지 좋음
날씨 실시간 기상 정보 확인하기 주간예보
"쓸 곳 많은 선물이 최고죠!"
선물은 N 네이버페이 기프트카드
Pay
책문화 함께N 여행+ 디자인 경제M JOB& 과학 중국 비즈니스 FARM
쇼핑 상품 쇼핑몰 MEN

■ 불법인가? 합법인가?

■ 합법이다!


제주도 렌트카 가격 비교와 예약을 한번에!
로그인
예약확인
공지사항
예약가능 차량 : 2,945개
최저가 : ₩3,900
최저가순
인기순
최저가 차량 리스트
회사
차량가
보험가
기타
올뉴모닝(휘)
₩74,000 / 24시간
₩ 3,900 / 24시간
실시간 예약 가능 차량 : 112
실시간 결제
경형
4
1000CC
보험선택안함
₩ 3,900 / 24시간
보험미선택 / 2일
₩ 4,200 / 24시간
₩ 5,000 / 24시간
₩ 5,400 / 24시간
₩ 5,900 / 24시간
₩ 6,000 / 24시간
₩ 6,800 / 24시간
₩ 6,800 / 24시간
₩ 6,800 / 24시간
₩ 7,600 / 24시간
₩ 7,600 / 24시간
₩ 8,000 / 24시간
₩ 8,500 / 24시간
₩ 8,500 / 24시간
₩ 10,300 / 24시간
₩ 16,000 / 24시간
₩ 16,800 / 24시간

■ 불법이다!



■ 불법 사례

대법원 "웹사이트 무단 크롤링은 불법"

김동훈 기자, 99re@bizwatch.co.kr
2017.09.27(수) 17:37

잡코리아, 사람인 상대 9년 소송전서 승소
크롤링(데이터 수집) 법적 기준 정립

리그베다위키 저작권 침해 사건서 DB제작자 권리 인정받아 최종 승소	2017-12-15

법무법인 민후는 서브컬처 위키백과 '리그베다위키(옛 엔하위키)' 운영자를 대리해 '엔하위키 미러' 운영자를 상대로 저작권 침해 및 부정경쟁방지법 위반 등에 기인한 형사고소 및 손해배상청구 소송을 제기하고 최종 승소했습니다.

■ 불법은 아니지만 조심하자

1. 스크랩하는 컨텐츠에 지적재산권이 있는지
2. 크롤링 하는 행동이 사이트에 큰 부담을 주지 않는지
3. 크롤러가 사이트의 이용방침을 위반하지 않는지
4. 크롤러가 사용자의 민감한 정보를 가져오지 않는지
5. 가져온 컨텐츠를 적합한 사용 표준하에 사용하는지

■ robots.txt

○ 웹사이트에 Crawler같은 (ro)bot들의 접근을 제어하기 위한 규약

○ 아직 권고안이라 꼭 지킬 의무는 없음

Basic format:

```
User-agent: [user-agent name]
Disallow: [URL string not to be crawled]
```

이름	User-Agent
Google	Googlebot
Google image	Googlebot-image
Msn	MSNBot
Naver	Yeti[2]
Daum	Daumoa

```
""
User-agent: *
Allow: /
""
```

```
""
User-agent: *
Disallow: /
""
```

■ 사용된 웹 기술 확인

○ builtwith

```
#!pip install builtwith

import builtwith

builtwith.parse('http://www.google.com')
builtwith.parse('http://www.facebook.com')
builtwith.parse('http://www.wordpress.com')
builtwith.parse('http://example.webscrapping.com')
```

■ 웹사이트 소유자 확인

○ whois

```
#!pip install python-whois

import whois

print(whois.whois("naver.com"))
print(whois.whois("appspot.com"))
```

■ 웹사이트 다운로드

○ urllib

```
import urllib
```

`urllib` is a package that collects several modules for working with URLs:

- `urllib.request` for opening and reading URLs
- `urllib.error` containing the exceptions raised by `urllib.request`
- `urllib.parse` for parsing URLs
- `urllib.robotparser` for parsing `robots.txt` files

○ request

The `urllib.request` module defines functions and classes which help in opening URLs (mostly HTTP) in a complex world — basic and digest authentication, redirections, cookies and more.

```
from urllib.request import urlopen
```

➢ urlopen

`urllib.request.`**`urlopen`**(*url*, *data=None*, [*timeout*,]*, *cafile=None*, *capath=None*, *cadefault=False*, *context=None*)

Open the URL *url*, which can be either a string or a `Request` object.

```
resp = urlopen("http://python.org")
html = resp.read()
```

○ response

The `urllib.response` module defines functions and classes which define a minimal file like interface, including `read()` and `readline()`. The typical response object is an addinfourl instance, which defines an `info()` method and that returns headers and a `geturl()` method that returns the url. Functions defined by this module are used internally by the `urllib.request` module.

➢ read

➢ info

○ error

The `urllib.error` module defines the exception classes for exceptions raised by `urllib.request`. The base exception class is `URLError`.

code

An HTTP status code as defined in **RFC 2616**. This numeric value corresponds to a value found in the dictionary of codes as found in `http.server.BaseHTTPRequestHandler.responses`.

reason

This is usually a string explaining the reason for this error.

headers

The HTTP response headers for the HTTP request that caused the `HTTPError`.

```
from urllib.error import HTTPError

try:
    resp = urlopen("http://www.google.com/search?q=korean")
except HTTPError as e:
    print("error:", e.code, e.reason, e.headers)
```

■ HTTP Errors

○ 4XX Client Errors

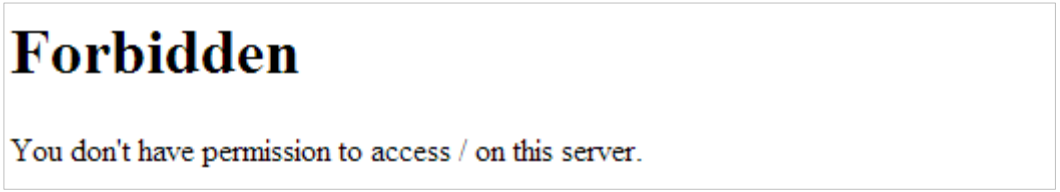
➢ This class of status code is intended for situations in which the error seems to have been caused by the client

○ 5XX Server Errors

➢ The server failed to fulfil a request

```
def download(url, num_retries=2):
    try:
        html = urlopen(url).read().decode("utf-8")
    except HTTPError as e:
        html = None

        print("error:", e.code, e.reason)

        if 500 <= e.code < 600 and num_retries>0:
            return download(url, num_retries-1)

    return html

#download("http://www.google.com/search?q=korean")
download("http://httpstat.us/500")
```

■ Header



○ Request

class `urllib.request.`**`Request`**(*url*, *data=None*, *headers={}*, *origin_req_host=None*, *unverifiable=False*, *method=None*)

This class is an abstraction of a URL request.

headers should be a dictionary, and will be treated as if `add_header()` was called with each key and value as arguments. This is often used to "spoof" the `User-Agent` header value, which is used by a browser to identify itself – some HTTP servers only allow requests coming from common browsers as opposed to scripts. For example, Mozilla Firefox may identify itself as `"Mozilla/5.0 (X11; U; Linux i686) Gecko/20071127 Firefox/2.0.0.11"`, while `urllib`'s default user agent string is `"Python-urllib/2.6"` (on Python 2.6).

○ 예제

```
from urllib.request import Request

def download(url, agent="python ", num_retries=2):
    headers = {'User-agent':agent}
    req = Request(url, headers=headers)

    try:
        html = urlopen(req).read().decode("utf-8")
    except HTTPError as e:
        html = None

        print("error:", e.code, e.reason)

        if 500 <= e.code < 600 and num_retries>0:
            return download(url, num_retries-1)

    return html

download("http://www.google.com/search?q=korean")
```

■ urllib vs Requests

See also: The Requests package is recommended for a higher-level HTTP client interface.

requests.**request**(*method, url, **kwargs*)

Constructs and sends a **Request**.

```python
import requests

def download(url, agent="python", num_retries=2):
    headers = {'User-agent':agent}

    req = requests.request("get", url, headers=headers)

    if 500 <= req.status_code < 600 and num_retries>0:
        print("error:", req.status_code, req.reason)
        return download(url, num_retries=(num_retries-1))

    return req

#html = download("http://www.google.com/search?q=korean")
html = download("http://httpstat.us/500")
print(html)
```

Parameters:

- **method** – method for the new **Request** object.
- **url** – URL for the new **Request** object.
- **params** – (optional) Dictionary or bytes to be sent in the query string for the **Request**.
- **data** – (optional) Dictionary or list of tuples `[(key, value)]` (will be form-encoded), bytes, or file-like object to send in the body of the **Request**.
- **json** – (optional) A JSON serializable Python object to send in the body of the **Request**.
- **headers** – (optional) Dictionary of HTTP Headers to send with the **Request**.
- **cookies** – (optional) Dict or CookieJar object to send with the **Request**.
- **files** – (optional) Dictionary of `'name': file-like-objects` (or `{'name': file-tuple}`) for multipart encoding upload. `file-tuple` can be a 2-tuple `('filename', fileobj)`, 3-tuple `('filename', fileobj, 'content_type')` or a 4-tuple `('filename', fileobj, 'content_type', custom_headers)`, where `'content-type'` is a string defining the content type of the given file and `custom_headers` a dict-like object containing additional headers to add for the file.
- **auth** – (optional) Auth tuple to enable Basic/Digest/Custom HTTP Auth.
- **timeout** (*float or tuple*) – (optional) How many seconds to wait for the server to send data before giving up, as a float, or a (connect timeout, read timeout) tuple.
- **allow_redirects** (*bool*) – (optional) Boolean. Enable/disable GET/OPTIONS/POST/PUT/PATCH/DELETE/HEAD redirection. Defaults to `True`.
- **proxies** – (optional) Dictionary mapping protocol to the URL of the proxy.
- **verify** – (optional) Either a boolean, in which case it controls whether we verify the server's TLS certificate, or a string, in which case it must be a path to a CA bundle to use. Defaults to `True`.
- **stream** – (optional) if `False`, the response content will be immediately downloaded.
- **cert** – (optional) if String, path to ssl client cert file (.pem). If Tuple, ('cert', 'key') pair.

Returns: **Response** object

■ urllib 한글

○ parse

```
from urllib import parse
```

This module defines a standard interface to break Uniform Resource Locator (URL) strings up in components (addressing scheme, network location, path etc.), to combine the components back into a URL string, and to convert a “relative URL” to an absolute URL given a “base URL.”

➢ parsing

`urllib.parse.`**`urlparse`**(*urlstring*, *scheme=''*, *allow_fragments=True*)

Parse a URL into six components, returning a 6-tuple. This corresponds to the general structure of a URL: `scheme://netloc/path;parameters?query#fragment`. Each tuple item is a string, possibly empty. The components are not broken up in smaller parts (for example, the network location is a single string), and % escapes are not expanded. The delimiters as shown above are not part of the result, except for a leading slash in the *path* component, which is retained if present. For example:

➢ quoting

`urllib.parse.`**`quote`**(*string*, *safe='/'*, *encoding=None*, *errors=None*)

Replace special characters in *string* using the `%xx` escape. Letters, digits, and the characters `'_.-~'` are never quoted. By default, this function is intended for quoting the path section of URL. The optional *safe* parameter specifies additional ASCII characters that should not be quoted — its default value is `'/'`.

○ 예제

```
parse.quote("한글")
```

```
from urllib import parse
from urllib.request import Request

def download(url, agent="python ", num_retries=2):
    print(url)
    headers = {'User-agent':agent}
    req = Request(url, headers=headers)

    try:
        html = urlopen(req).read().decode("utf-8")
    except HTTPError as e:
        html = None

        print("error:", e.code, e.reason)

        if 500 <= e.code < 600 and num_retries>0:
            return download(url, num_retries-1)

    return html

html = download("http://www.google.com/search?q="+parse.quote("한글")+"&ie=utf-8&oe=utf-8")
#html = download("http://search.naver.com/search.naver?query="+parse.quote("한글"))
html
```

○ 예제

```
import requests

def download(url, agent="python", num_retries=2):
    headers = {'User-agent':agent}

    req = requests.request("get", url, headers=headers)

    if 500 <= req.status_code < 600 and num_retries>0:
        print("error:", req.status_code, req.reason)
        return download(url, num_retries=(num_retries-1))

    return req

html = download("http://www.google.com/search?q=한글&ie=utf-8&oe=utf-8")

print(html.encoding)

html.encoding = "utf-8"

print(type(html.text))
print(type(html.content))

html.text
```

○ get / post

`requests.get`(*url*, *params=None*, ***kwargs*) [source]

Sends a GET request.

Parameters:
- **url** – URL for the new `Request` object.
- **params** – (optional) Dictionary or bytes to be sent in the query string for the `Request`.
- ****kwargs** – Optional arguments that `request` takes.

Returns: `Response` object

Return type: requests.Response

`requests.post`(*url*, *data=None*, *json=None*, ***kwargs*) [source]

Sends a POST request.

Parameters:
- **url** – URL for the new `Request` object.
- **data** – (optional) Dictionary (will be form-encoded), bytes, or file-like object to send in the body of the `Request`.
- **json** – (optional) json data to send in the body of the `Request`.
- ****kwargs** – Optional arguments that `request` takes.

Returns: `Response` object

Return type: requests.Response

○ 예제

➢ get

```
import requests

def download(url, agent="python", num_retries=2):
    headers = {'User-agent':agent}

    req = requests.get(url, params={"q":"한글", "ie":"utf-8", "oe":"utf-8"}, headers=headers)

    if 500 <= req.status_code < 600 and num_retries>0:
        print("error:", req.status_code, req.reason)
        return download(url, num_retries=(num_retries-1))

    return req

html = download("http://www.google.com/search")

html.url
```

○ 예제

➢ post - dict

```
import requests

def download(url, agent="python", num_retries=2):
    headers = {'User-agent':agent}

    req = requests.post(url, data={"q":"한글", "ie":"utf-8", "oe":"utf-8"}, headers=headers)

    if 500 <= req.status_code < 600 and num_retries>0:
        print("error:", req.status_code, req.reason)
        return download(url, num_retries=(num_retries-1))

    return req

html = download("http://httpbin.org/post")

print(html.url)
print(html.text)
```

○ 예제

➢ post - json

```
import json
import requests

def download(url, agent="python", num_retries=2):
    headers = {'User-agent':agent}

    req = requests.post(url, json=json.dumps({"q":"한글", "ie":"utf-8", "oe":"utf-8"}), headers=headers)

    if 500 <= req.status_code < 600 and num_retries>0:
        print("error:", req.status_code, req.reason)
        return download(url, num_retries=(num_retries-1))

    return req

html = download("http://httpbin.org/post")

print(html.url)
print(html.text)
```

1.2. Crawling

■ What is Crawling

○ Web crawlers are known by a variety of names – spiders, bots or web crawlers

○ Systematically browses the World Wide Web

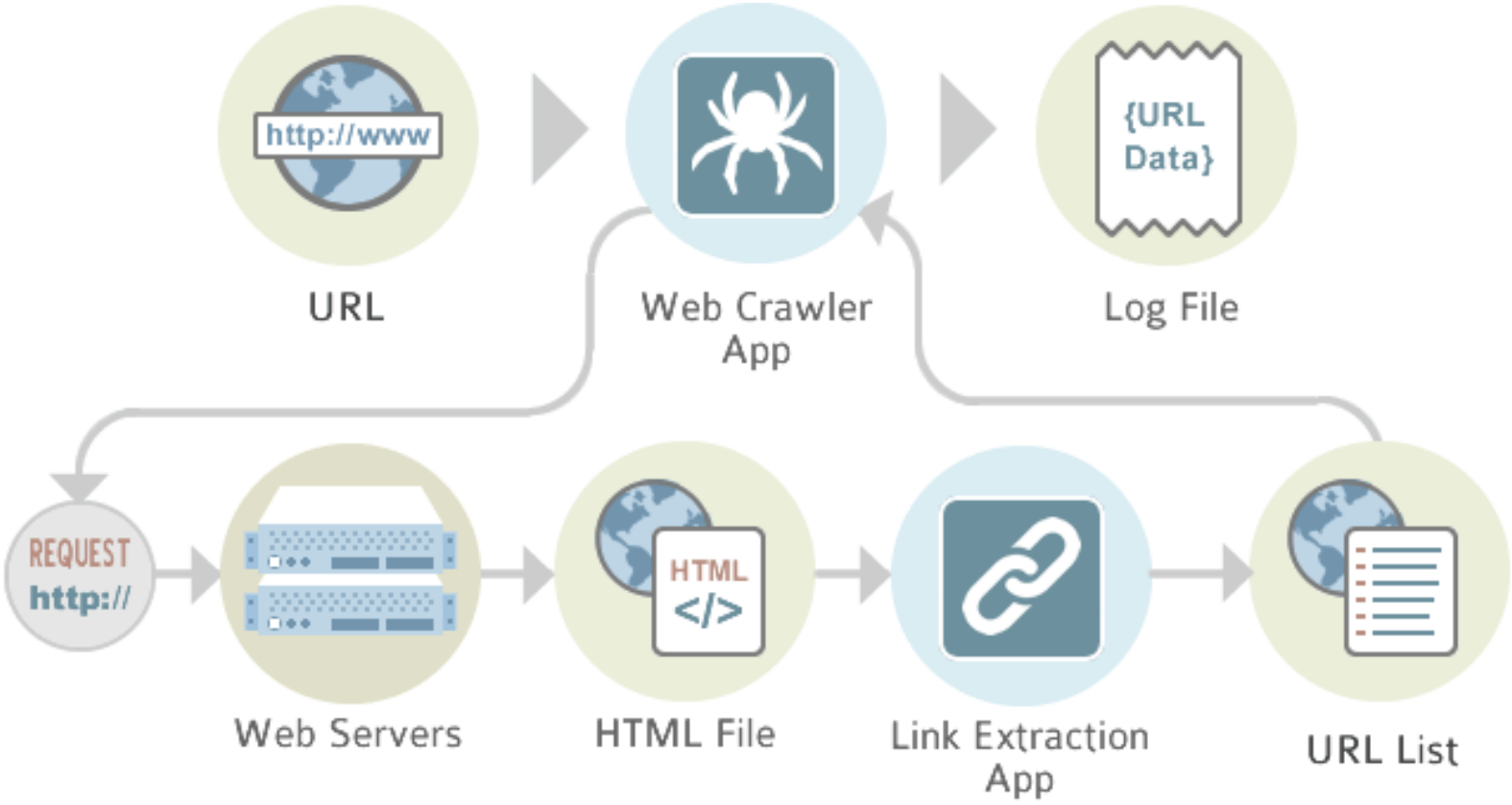
■ Why use Crawling

○ Typically for the purpose of Web indexing

○ Specifically, they index what words are used on a website and in what context

■ Crawling Architecture

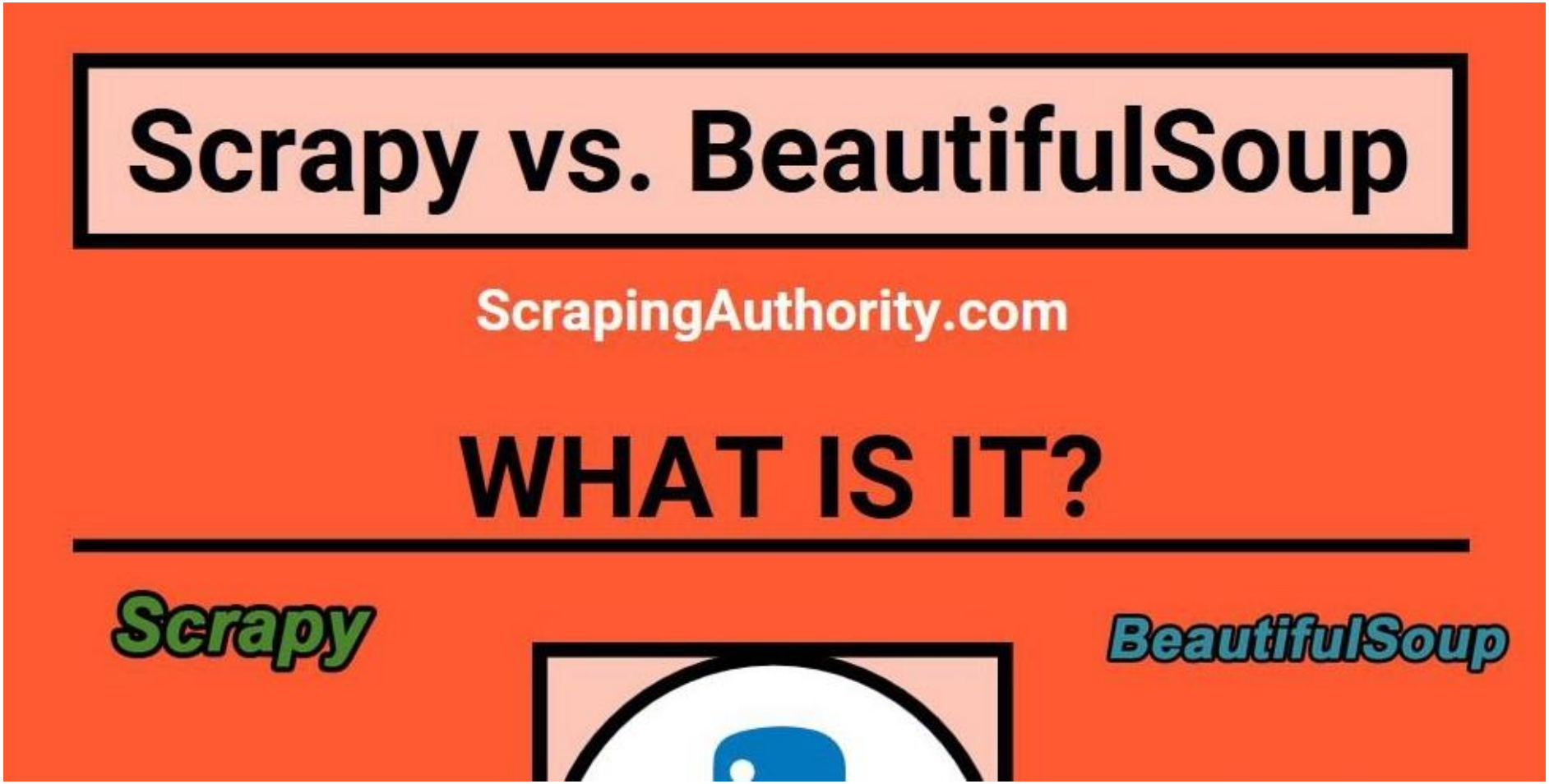
○ Process of repetitively finding and fetching hyperlinks starting from a list of starting URLs



■ Scrapy vs BeautifulSoup

○ Scrapy is a Web-spider or web scraper framework

○ BeautifulSoup is a parsing library



■ What is bs4?

○ Beautiful Soup is a Python library for pulling data out of HTML and XML files

■ bs4

```
!pip install beautifulsoup4
```

```
from bs4 import BeautifulSoup
```

Parser	Typical usage	Advantages	Disadvantages
Python's html.parser	`BeautifulSoup(markup, "html.parser")`	• Batteries included • Decent speed • Lenient (as of Python 2.7.3 and 3.2.)	• Not very lenient (before Python 2.7.3 or 3.2.2)
lxml's HTML parser	`BeautifulSoup(markup, "lxml")`	• Very fast • Lenient	• External C dependency
lxml's XML parser	`BeautifulSoup(markup, "lxml-xml")` `BeautifulSoup(markup, "xml")`	• Very fast • The only currently supported XML parser	• External C dependency
html5lib	`BeautifulSoup(markup, "html5lib")`	• Extremely lenient • Parses pages the same way a web browser does • Creates valid HTML5	• Very slow • External Python dependency

➢ If you can, I recommend you install and use lxml for speed. If you're using a version of Python 2 earlier than 2.7.3, or a version of Python 3 earlier than 3.2.2, it's essential that you install lxml or html5lib–Python's built-in HTML parser is just not very good in older versions

■ 변경사항

- `renderContents` -> `encode_contents`
- `replaceWith` -> `replace_with`
- `replaceWithChildren` -> `unwrap`
- `findAll` -> `find_all`
- `findAllNext` -> `find_all_next`
- `findAllPrevious` -> `find_all_previous`
- `findNext` -> `find_next`
- `findNextSibling` -> `find_next_sibling`
- `findNextSiblings` -> `find_next_siblings`
- `findParent` -> `find_parent`
- `findParents` -> `find_parents`
- `findPrevious` -> `find_previous`
- `findPreviousSibling` -> `find_previous_sibling`
- `findPreviousSiblings` -> `find_previous_siblings`
- `nextSibling` -> `next_sibling`
- `previousSibling` -> `previous_sibling`

○ BeautifulSoup

➢ Beautiful Soup transforms a complex HTML document into a complex tree of Python objects

○ Tag

➢ A Tag object corresponds to an XML or HTML tag in the original document

○ Name

➢ Every tag has a name, accessible as ***.name***

○ Attributes

➢ A tag may have any number of attributes. The tag ***<b id="boldest">*** has an attribute ***"id"*** whose value is ***"boldest"***, accessible as ***Tag['id']***

○ 예제

```
html = """
    <html>
        <head></head>
        <body>
            <div id="wrap">
                <p class="content">
                    <a href="_blank">Click</a>
                </p>
            </div>
        </body>
    </html>
"""
```

```
doc = BeautifulSoup(html, "lxml")
```

```
doc.div["id"]
doc.p["class"]
doc.a["href"]
doc.a.attrs
```

○ find

Signature: find(name, attrs, recursive, string, **kwargs)

The `find_all()` method scans the entire document looking for results, but sometimes you only want to find one result.

○ find_all

Signature: find_all(name, attrs, recursive, string, limit, **kwargs)

The `find_all()` method looks through a tag's descendants and retrieves *all* descendants that match your filters.

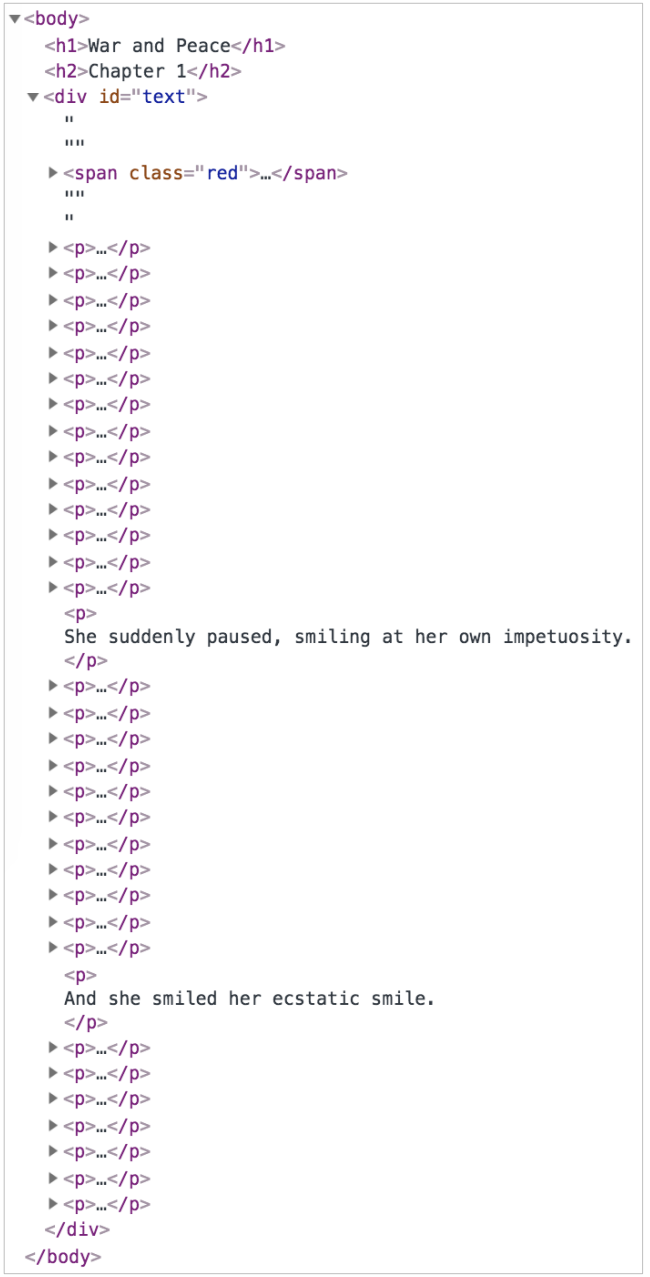
○ get_text

If you only want the text part of a document or tag, you can use the `get_text()` method.
It returns all the text in a document or beneath a tag, as a single Unicode string:

○ 예제

➢ http://www.pythonscraping.com/pages/warandpeace.html

```
html = requests.get("http://www.pythonscraping.com/pages/
doc = BeautifulSoup(html.text, "lxml")

div = doc.find("div", id="text")
print(div.get_text())

#tagList = doc.find_all("span", class="green")
tagList = doc.find_all("span", class_="green")
print(len(tagList))

for tag in tagList:
    print(tag.get_text())

tagList = doc.find_all("span", {"class":"green"})
print(len(tagList))

for tag in tagList:
    print(tag.get_text())
```



○ 부모 탐색

- find_parent
  - You can access an element's parent with the ***.parent*** attribute
- find_parents
  - You can iterate over all of an element's parents with ***.parents***

○ 자식 탐색

- find_all
  - The ***.children*** attributes only consider a tag's direct children
  - The ***.descendants*** attribute lets you iterate over all of a tag's children, recursively
- find_all(recursive=False)
  - you can iterate over a tag's children using the ***.children*** generator

○ 형제 탐색

- find_next_sibling, find_previous_sibling
  - navigate between page elements that are on the same level of the parse tree
- find_next_siblings, find_previous_siblings

○ 예제

➢ http://www.pythonscraping.com/pages/page3.html

```
<body>
 <div id="wrapper">
   <img src="../img/gifts/logo.jpg" style="float:left;">
   <h1>Totally Normal Gifts</h1>
  <div id="content">
    "Here is a collection of totally normal, totally reasonable gifts that your friends are sure to love! Our collection is
    hand-curated by well-paid, free-range Tibetan monks."
   <p>…</p>
  </div>
  <table id="giftList">
   <tbody>
    <tr>
      <th>
      Item Title
      </th>
      <th>
      Description
      </th>
      <th>
      Cost
      </th>
      <th>
      Image
      </th>
    </tr>
    <tr id="gift1" class="gift">…</tr>
    <tr id="gift2" class="gift">…</tr>
    <tr id="gift3" class="gift">…</tr>
    <tr id="gift4" class="gift">…</tr>
    <tr id="gift5" class="gift">…</tr>
   </tbody>
  </table>
  <p></p>
  <div id="footer">…</div>
 </div>
</body>
```

➢ difference between tag and find/find_all

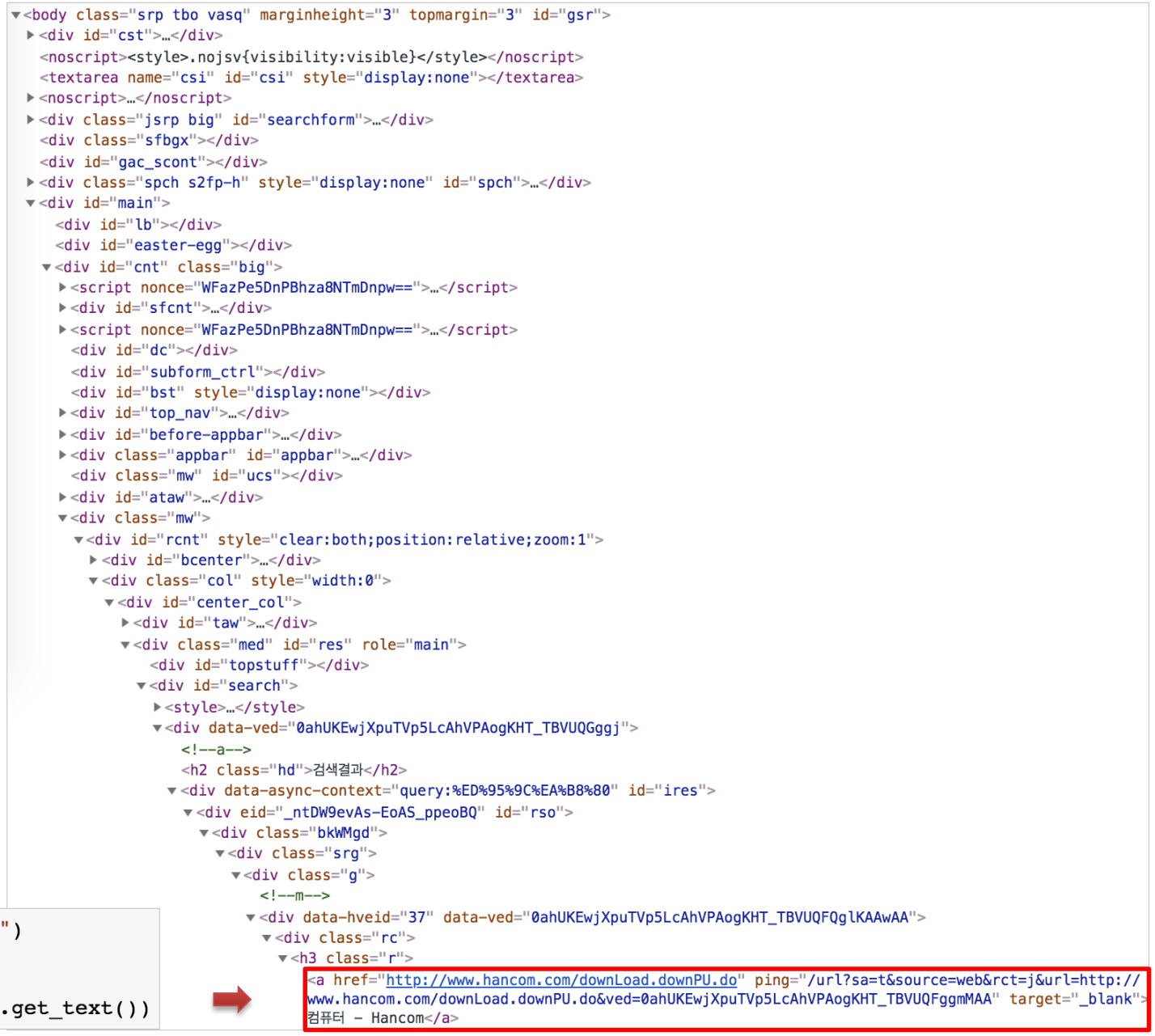
```
html = requests.get("http://www.pythonscraping.com/pages/page3.html")
doc = BeautifulSoup(html.text, "lxml")

div1 = doc.div
print(type(div1))

div2 = doc.find_all("div")
print(type(div2))
for tag in div2:
    print(type(tag))
```

```
node = doc.find("div", {"id":"footer"})

parent = node.find_parent()

parents = node.find_parents()

children = parent.find_all(recursive=False)

descendants = parent.find_all()

prevsibling = children[1].find_previous_sibling()

nextsibling = children[1].find_next_sibling()
```

■ CSS Selector

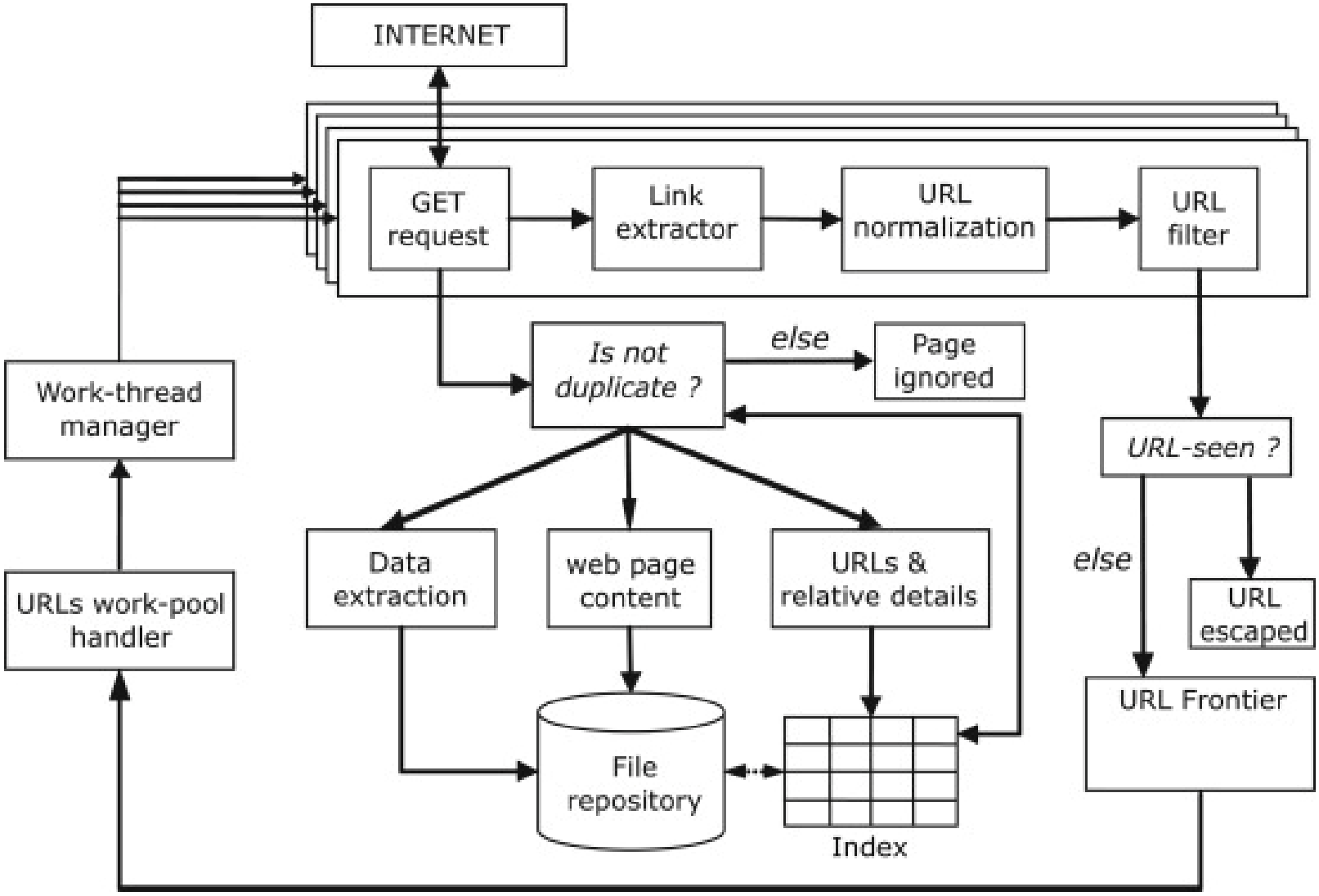
○ select

➢ Beautiful Soup supports the most commonly-used CSS selectors

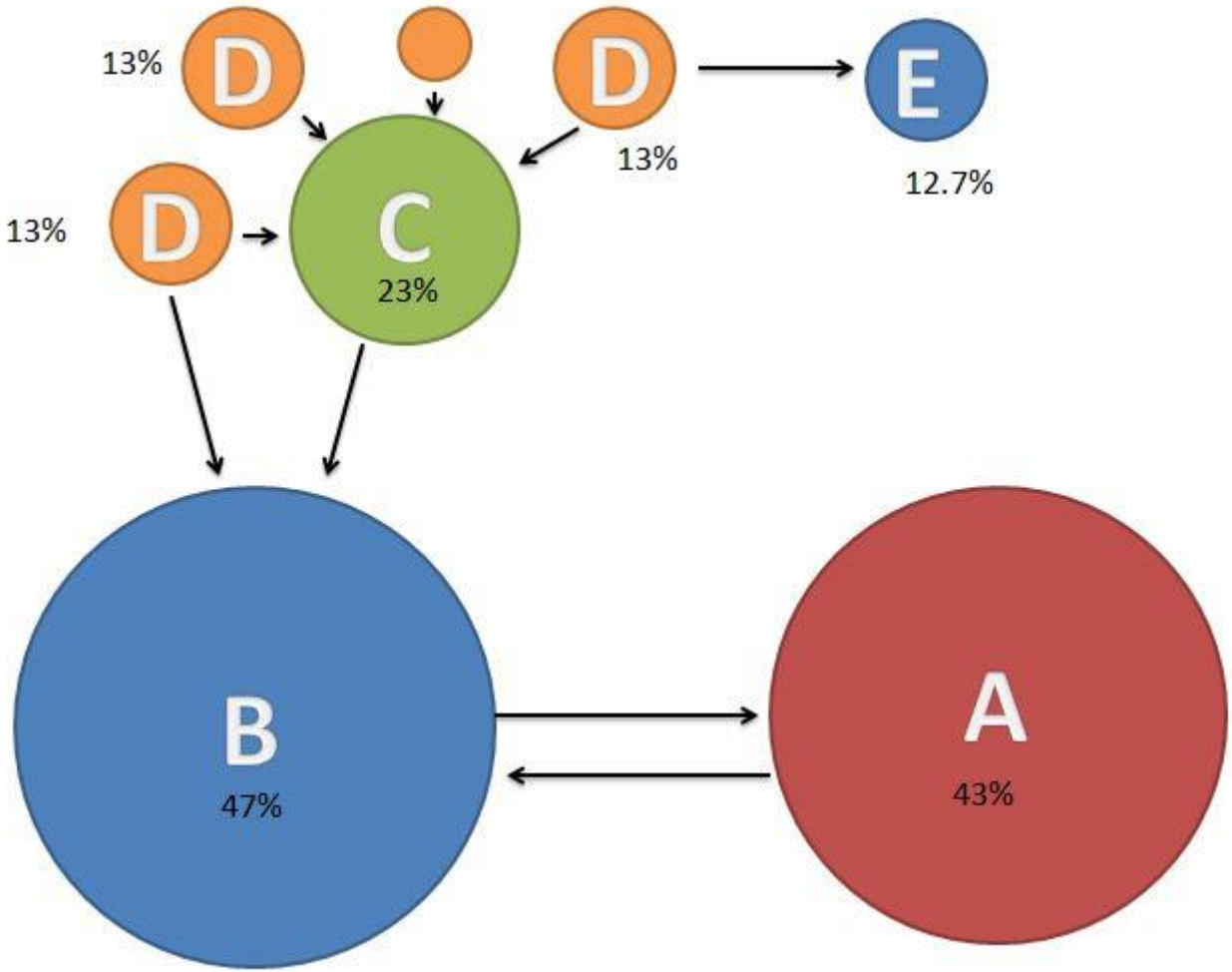



```
links = doc.select("h3.r > a")
print(len(links))
for link in links:
    print(link["href"], link.get_text())
```

■ Links Crawling



■ Page Rank

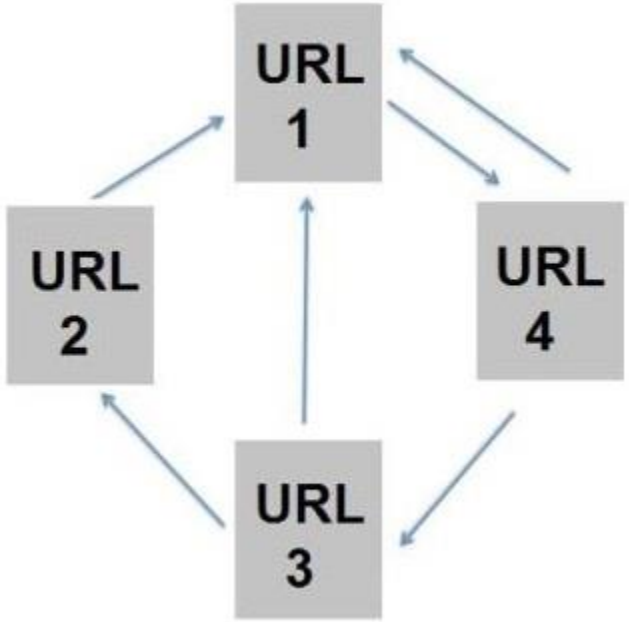


$$PageRank\ of\ site = \sum \frac{PageRank\ of\ inbound\ link}{Number\ of\ links\ on\ that\ page}$$

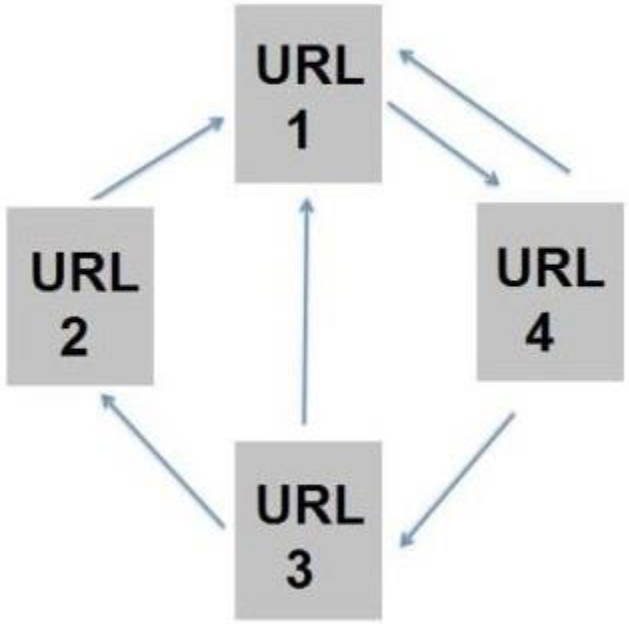
OR

$$PR(u) = (1-d) + d \times \sum \frac{PR(v)}{N(v)}$$

url_1 url_4
url_2 url_1
url_3 url_2
url_3 url_1
url_4 url_3
url_4 url_1



url_1	url_4
url_2	url_1
url_3	url_2
url_3	url_1
url_4	url_3
url_4	url_1



$$A^{12}v = \begin{pmatrix} 0.366 \\ 0.089 \\ 0.184 \\ 0.360 \end{pmatrix}$$

iterative

$$A = \begin{pmatrix} 0 & 1 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & \frac{1}{2} \\ 1 & 0 & 0 & 0 \end{pmatrix} \quad v = \begin{pmatrix} 0.25 \\ 0.25 \\ 0.25 \\ 0.25 \end{pmatrix} = \begin{pmatrix} 0.5 \\ 0.125 \\ 0.125 \\ 0.25 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 1 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & \frac{1}{2} \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0.5 \\ 0.125 \\ 0.125 \\ 0.25 \end{pmatrix} = \begin{pmatrix} 0.312 \\ 0.062 \\ 0.125 \\ 0.5 \end{pmatrix}$$

■ Collecting Links

```
def getUrls(url, params=None, num_retries=2):
    headers = {"User-agent":"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_5) AppleWebKit/537.36

    resp = requests.get(url, params=params, headers=headers)

    if 500 <= resp.status_code < 600 and num_retries>0:
        print("error:", resp.status_code, resp.reason)
        return getUrls(url=url, params=params, num_retries=(num_retries-1))

    html = BeautifulSoup(resp.content, "lxml")
    links = html.select("h3.r > a")

    return [link["href"] for link in links if link.has_attr("href") == True]



seed = "https://www.google.co.kr/search"
params = {
    "q":"한글",
    "ie":"utf-8"
}

urlList = getUrls(seed, params)
print(urlList)
```

■ Following Links

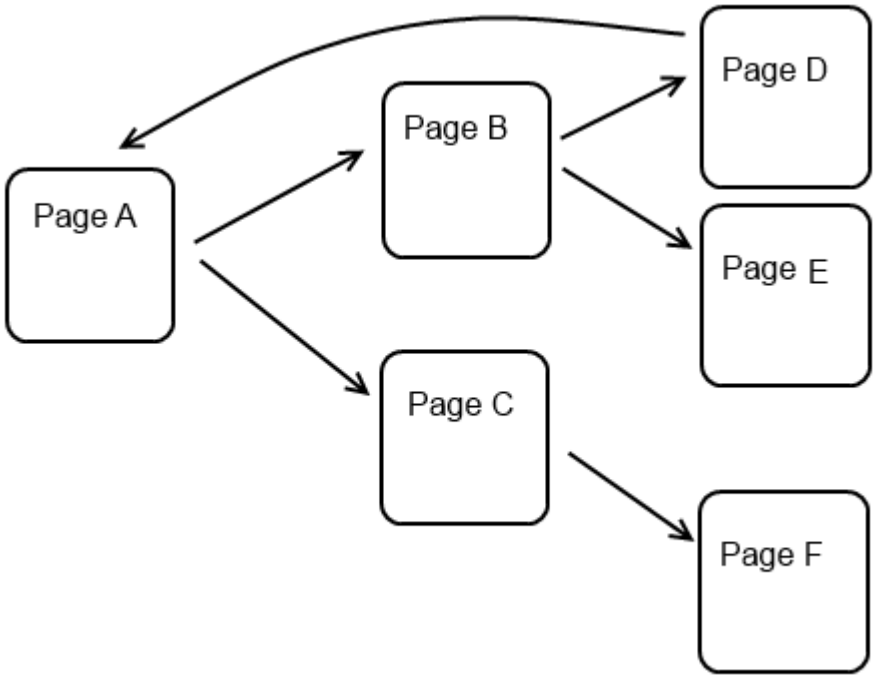
```
while queue:
    url = queue.pop()
    urlList = getUrls(url)
    print(url, len(urlList))
    queue.extend(urlList)
```

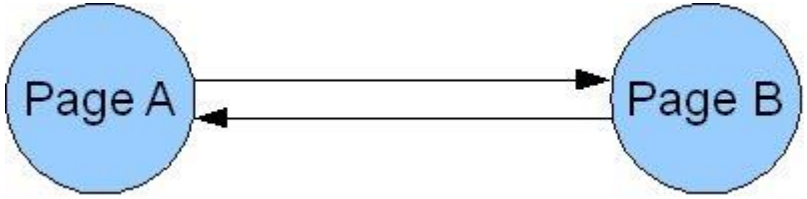
○ 예제

```
def getUrls(url, params=None, select="a", num_retries=2):
    headers = {"User-agent":"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_5) AppleWebKit/537.36 (KHTML, like Gecko)

    resp = requests.get(url, params=params, headers=headers)

    if 500 <= resp.status_code < 600 and num_retries>0:
        print("error:", resp.status_code, resp.reason)
        return getUrls(url=url, params=params, num_retries=(num_retries-1))

    html = BeautifulSoup(resp.content, "lxml")
    links = html.select(select)

    return [link.get("href") for link in links if link.has_attr("href") == True]

def checkUrl(url):
    dest = parse.urlparse(url)

    if len(dest.scheme) > 0 and dest.scheme in ["http", "https"]:
        return True
    else:
        return False

seed = "https://www.google.com/search"
params = {
    "q":"한글",
    "ie":"utf-8"
}

queue = getUrls(seed, params, "h3.r > a")
result = list()

while queue:
    url = queue.pop()

    while checkUrl(url) is False:
        url = queue.pop()

    urlList = getUrls(url)

    result.append(url)
    result.extend(urlList)

    print(url, len(urlList), end="\n\n")
```

■ Circular Links

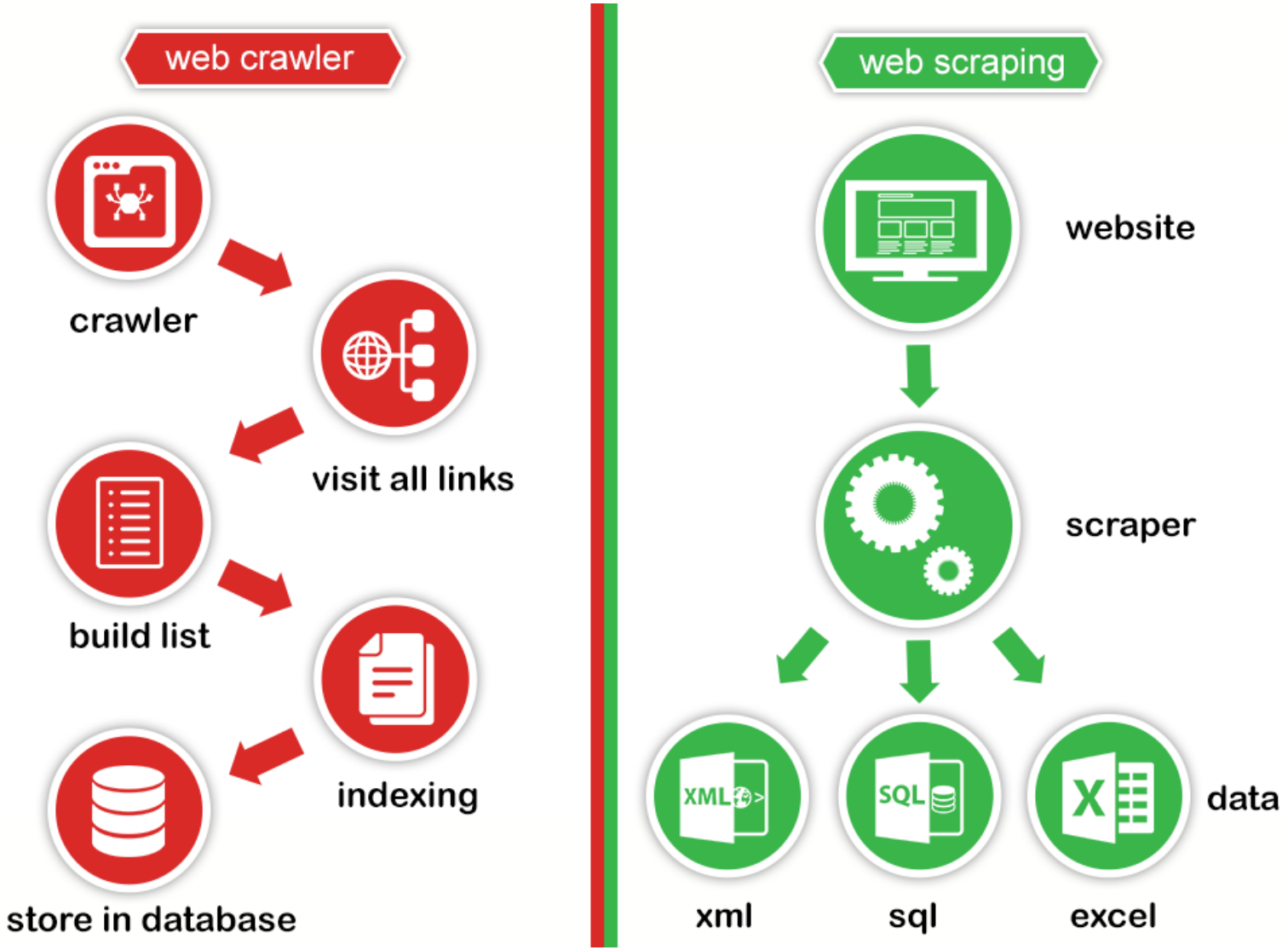
도메인,
방문이력





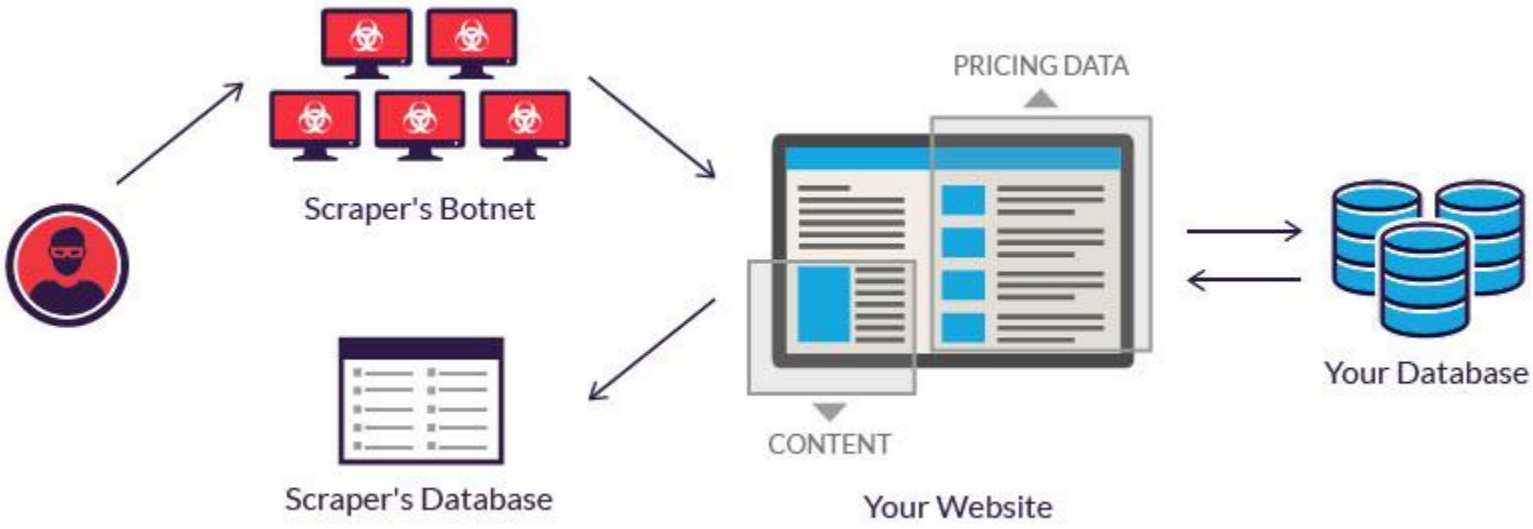
1.3. Scraping

■ Crawling vs Scraping



■ Scraping Architecture

○ Process of automatically requesting a web document and collecting information from it



■ 자유게시판 스크래핑

○ 뽐뿌

```
seed = "http://www.ppomppu.co.kr/zboard/zboard.php"
params = {
    "id":"freeboard",
    "page":"1"
}

headers = {"User-agent":"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_5) AppleWebKit/537.36 (KHTML, like Gecko)

resp = requests.get(seed, params=params, headers=headers)
resp.content
```

○ HTML Validator

- https://validator.w3.org/
- https://validator.w3.org/unicorn/?ucn_lang=ko

Result:	1340 Errors, 341 warning(s)

○ urllib

```
seed = "http://www.ppomppu.co.kr/zboard/zboard.php?"
params = {
    "id":"freeboard",
    "page":"1"
}

headers = {"User-agent":"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_5) AppleWebKit/537.36 (KHTML, like Gecko)

print(parse.urlencode(params))

req = Request(seed+parse.urlencode(params), headers=headers)
res = urlopen(req).read()
type(res)

res.decode("euc-kr")
```

○ How to select?

```
▼<table id="revolution_main_table" border="0" cellspacing="0" cellpadding="0" width="900" class="title_bg">
  ▶<colgroup>…</colgroup>
    <form method="post" name="list" action="list_all.php"></form>
    <input type="hidden" name="page" value="1">
    <input type="hidden" name="id" value="freeboard">
    <input type="hidden" name="select_arrange" value="headnum">
    <input type="hidden" name="desc" value="asc">
    <input type="hidden" name="page_num" value="30">
    <input type="hidden" name="selected">
    <input type="hidden" name="exec">
    <input type="hidden" name="keyword" value>
    <input type="hidden" name="sn" value="off">
    <input type="hidden" name="ss" value="Array">
    <input type="hidden" name="sc" value="off">
    <input type="hidden" name="html">
  ▼<tbody>
    ▶<tr>…</tr>
    ▶<tr align="center" valign="middle" class="title_bg" style="height:27px;">…</tr>
    ▶<tr>…</tr>
    ▶<tr id="page_show_noti_1" name="page_show_noti_1" align="center" class="list_notice" onmouseover=
    "this.style.backgroundColor='#F5F5F5'" onmouseout="this.style.backgroundColor=''" style="word-break: break-all; display: table-
    row;" valign="middle">…</tr>
    ▶<tr>…</tr>
    ▼<tr align="center" class="list1" onmouseover="this.style.backgroundColor='#F5F5F5'" onmouseout="this.style.backgroundColor=''"
    style="word-break: break-all;" valign="middle">
        <td class="eng list_vspace" colspan="2">4780973</td>
        <!--<td class='han4 list_vspace' nowrap colspan=2><nobr class='han4 list_vspace'> </nobr></td>-->
        <!--<td nowrap colspan=2 style='padding:0'><input type=checkbox name=cart value="5942287"></td>-->
      ▶<td colspan="2" class="list_vspace" align="left">…</td>
      ▼<td align="left" class="list_vspace">
          <img src="/images/icon_04.png" border="0" align="absmiddle">
          "  "
        ▼<a href="view.php?id=freeboard&page=1&divpage=1109&no=5942287">
            <font class="list_title">※ 근무시간,장소 상관없는 재택알바 하실분 구합니다</font>
```

○ 목록 가져오기(urllib)

```
import re

def getContent(url, path):
    headers = {"User-agent":"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_5) AppleWebKit/537.36 (KHTML, like Gecko)

    req = Request(url, headers=headers)
    res = urlopen(req).read().decode("euc-kr", "ignore").encode("utf-8")

    html = BeautifulSoup(res, "html.parser")
    return html.select(path)


seed = "http://www.ppomppu.co.kr/zboard/"
params = {
    "id":"freeboard",
    "page":"1",
    "prev_page":None,
    "divpage":"1109"
}

tagList = getContent(seed+"zboard.php?"+parse.urlencode(params), "td.list_vspace > img + a")
#re.findall("<a.*href=\"view.php?(.*?)\".*><font.*>(.*?)</font></a>", res)

for tag in tagList[1:]:
    print(tag.get("href"), tag.get("href")[-7:], tag.get_text())
```

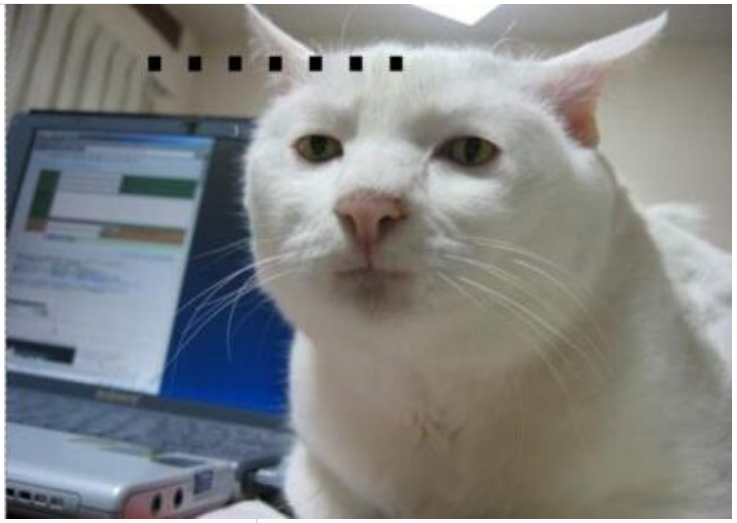
○ 목록 가져오기(requests)

```
def getUrls(url, params=None, num_retries=2):
    headers = {
        "User-agent":"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_5) AppleWebKit/537.36 (KHTML, like Gecko)
        "Accept-Encoding":"none"
    }

    resp = requests.get(url, params=params, headers=headers)

    if 500 <= resp.status_code < 600 and num_retries>0:
        print("error:", resp.status_code, resp.reason)
        return getUrls(url=url, params=params, num_retries=(num_retries-1))

    html = BeautifulSoup(resp.content, "lxml")
    links = html.select("td.list_vspace > a")

    return [link["href"] for link in links if link.has_attr("href") == True]



seed = "http://www.ppomppu.co.kr/zboard/zboard.php"
params = {
    "id":"freeboard",
    "page":"1"
}

urlList = getUrls(seed, params)
for row in urlList[1:]:
    print(row, row[-7:])
```

○ 갈수록 산

➢ feat. 본문 가져오기

```
<td class='board-contents' align="left" valign=top class=han>
<br />
최상층이라 옥탑 및 옥상이 있는 아파트입니다.<br />
<br />
낮에 잠깐나갔다가발견한건데 집거미로 추정되는 거미와 거미집들이 보이더라구요<br />
<br />
<br />
특히나 집에 집거미 작은거 간간히보여서 노이로제인데..<br />
<br />
뭐로없앨까요?<br />
<br />
토치는 없어서 불로구워버릴순없고..<br />
<br />
가스 헤어스프레이는 있는데 이걸로 불붙히자니 좀 위험하고..<br />
<br />
<br />
락스뿌리면죽으려나요..<br />
<br />
콘크리트외벽에 생긴거라 불이딱인데 토치가 아쉽네요..<br />
<br />
<br />
흑 무서움 ㅠ<!--"&lt;--></td></tr></table><!--/DCM_BODY--></td></tr></tbody></table>
```



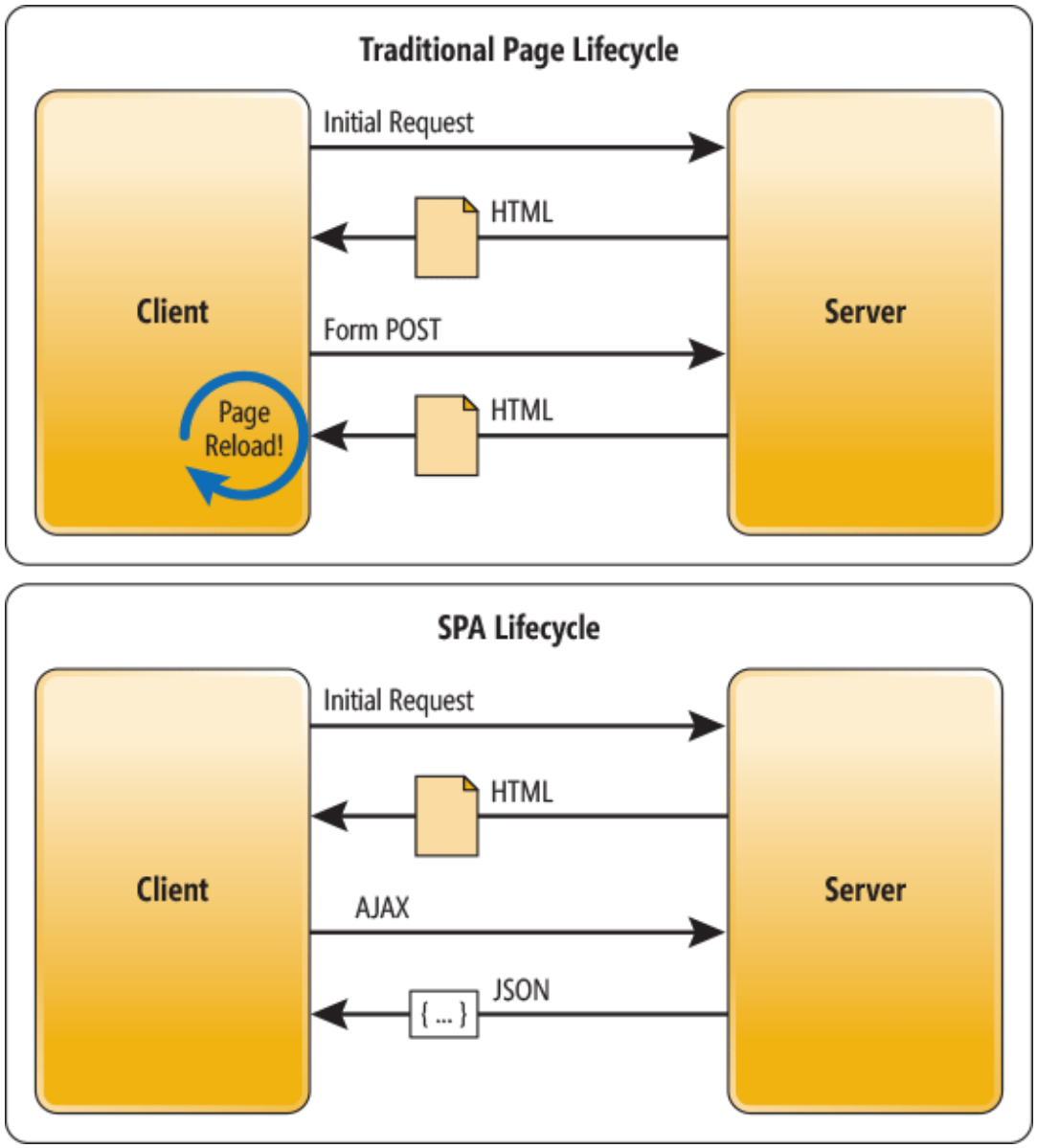
○ 본문 가져오기

```
import re

def getContent(url, path):
    headers = {"User-agent":"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_5) AppleWebKit/537.36 (KHTML, like Gecko)

    req = Request(url, headers=headers)
    res = urlopen(req).read().decode("euc-kr", "ignore").encode("utf-8")

    html = BeautifulSoup(res, "html.parser")
    return html.select(path)


seed = "http://www.ppomppu.co.kr/zboard/"
params = {
    "id":"freeboard",
    "page":"1",
    "prev_page":None,
    "divpage":"1109"
}

tagList = getContent(seed+"zboard.php?"+parse.urlencode(params), "td.list_vspace > img + a")
#re.findall("<a.*href=\"view.php?(.*?)\".*><font.*>(.*?)</font></a>", res)

for tag in tagList[1:]:
    print(tag.get("href"), tag.get("href")[-7:], tag.get_text())

    contentList = getContent(seed+tag.get("href"), "table.pic_bg table td.han")
    print(contentList[0].get_text().strip(), end="\n\n")
```

○ 댓글은 어떻게 가져와야 할까? (실습)

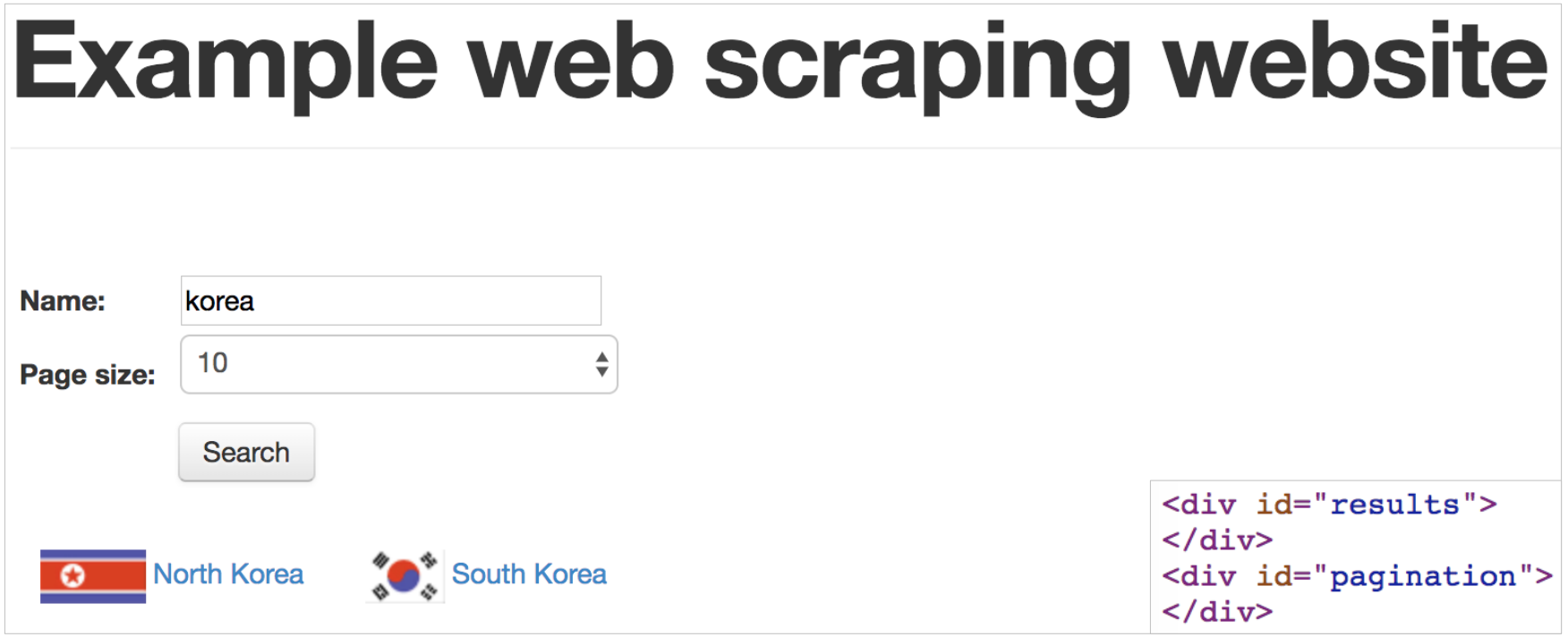
```
▶<table border="0" cellspacing="0" cellpadding="0" height="5" width="900">…</table>
▶<script type="text/JavaScript">…</script>
▼<div id="newbbs">
  ▼<div id="quote">
     <meta http-equiv="Content-Type" content="text/html; charset=euc-kr">
    ▶<script type="text/javascript">…</script>
    ▼<div id="iC_60812908" align="right" style="width: 880px; padding-bottom: 8px;">
      ▼<div id="comment_60812908" style="position:relative; overflow:hidden; width:100%;">
        ▼<div style="width:100%;height:auto;" align="left">
           <a name="60812908" id="60812908"></a>
           <p style="margin:8px 0;border-bottom:1px solid #E0E0E0;"></p>
          ▼<table border="0" width="100%" cellspacing="0" cellpadding="0" class="info_bg" style="table-layout:fixed;
          padding:0 0 0 0">
            ▼<tbody>
              ▼<tr valign="top">
                ▶<td width="100px" align="right" style="padding:0 0 0 0">…</td>
                 <td class="info_bg" width="6px"></td>
                 <td class="separator2" width="3px"></td>
                 <td width="3px"></td>
                ▼<td style=" padding:2px 4px 4px 5px;word-break:break-all;">
                  ▶<span class="cmd_box" onclick="anti_vote('freeboard', 5943955, 60812908);" style="float:right;
                  padding:0px 5px; margin-left:3px; cursor:pointer;  ">…</span>
                  ▶<span class="cmd_box" onclick="c_vote('freeboard', 5943955, 60812908);" style="float:right; padding:
                  0px 5px; cursor:pointer; ">…</span>
                   <img src="/images/phone_icon.png" style="padding-bottom:5px;">
                   " "
                  ▼<b>
                     <a href="#" onclick="javascript:zb_layer('', '', 'KPxKocMGS2Nm1xn8yQfsYA%3D%3D', 'freeboard',
                     '%C1%B7%B1%B8%C7%CF%B6%F3%B1%D7%B7%A1%B6%F3', '', '', '', '1', '1', '5943955', event); return false;
                     ">족구하라그래라</a>
                   </b>
                   <br>
                   <div id="commentContent_60812908" class="han" style="position:relative; overflow:hidden; width:100%;">
                   술쳐마셧구나</div>
```

■ Dynamic HTML / AJAX



○ 예제

➢ http://example.webscraping.com/places/default/search




```
resp = requests.get("http://example.webscraping.com/places/default/search")

html = BeautifulSoup(resp.content, "lxml")

result = html.find("#results")

print(result)
```

Selenium

■ What is Selenium?

○ Selenium automates browsers

○ Automating web applications for testing purposes

■ Why use Selenium

○ Rapid feedback to developers

○ Finding defects missed by manual testing

○ Support for Agile and extreme development methodologies

■ Another purpose

○ FORM 처리

○ AJAX Rendering 결과 획득

■ 설치 및 실행

```
!pip install selenium
```

```
from selenium import webdriver
```

```
driver = webdriver.Chrome()
```

```
webdriver.ActionChains
webdriver.Android
webdriver.BlackBerry
webdriver.Chrome
webdriver.ChromeOptions
webdriver.DesiredCapabilities
webdriver.Edge
webdriver.Firefox
webdriver.FirefoxOptions
webdriver.FirefoxProfile
```

```
!brew cas
```

```
==> Satis
==> Downl                                    orage.googleapis.com/2.40/chromedriver_ma
########                                     ################################## 100.0%
==> Verif                                    river
==> Insta
==> Extra                                    river
==> Linki                                    usr/local/bin/chromedriver'.
🍺  chrom                                    alled!
```

○ 예제

```
driver.get("http://example.webscraping.com/places/default/search")
```

```
driver.find_element_by_id("search_term").send_keys("korea")
```

```
driver.find_element_by_id("search").click()
```

```
driver.implicitly_wait(1)
```

```
links = driver.find_elements_by_css_selector("#results a")
```

```
[link.text for link in links]
```

```
['North Korea', 'South Korea']
```

○ find element(s)

- *find_element_by_id*
- *find_element_by_name*
- *find_element_by_xpath*
- *find_element_by_link_text*
- *find_element_by_partial_link_text*
- *find_element_by_tag_name*
- *find_element_by_class_name*
- *find_element_by_css_selector*

To find multiple elements (these methods will return a list):

- *find_elements_by_name*
- *find_elements_by_xpath*
- *find_elements_by_link_text*
- *find_elements_by_partial_link_text*
- *find_elements_by_tag_name*
- *find_elements_by_class_name*
- *find_elements_by_css_selector*

○ 네이버 로그인하기

```
import json
from selenium.common.exceptions import NoSuchElementException

with open("account.json", "r") as f:
    account = json.load(f)

driver.get("https://nid.naver.com/nidlogin.login?url=http%3A%2F%2Fmail.naver.com%2F")

driver.find_element_by_id("id").send_keys(account["id"])
driver.find_element_by_id("pw").send_keys(account["pw"])
driver.find_element_by_css_selector(".btn_global").click()
```

○ 네이버 로그인하기 – 보안설정 넘어가기

```
try:
    checkLogin = driver.find_element_by_class_name("btn_upload")

    driver.find_element_by_css_selector(".btn_upload > a").click()
    driver.find_element_by_class_name("btn_maintain").click()
except NoSuchElementException as e:
    print(e)
finally:
    html = BeautifulSoup(driver.page_source, "lxml")
```

○ 메일 주소 가져오기 (실습)

```
▼<div id="list_for_view" class="listWrap" style="top: 0px; font-size: 9pt;">
  <h4 class="blind">받은메일함 목록 - 시간순 보기 - 메일 목록만 보기</h4>
 ▼<ol class="mailList sender_context" style="min-width:600px">
  ▼<li class="7027_li  notRead unmark  _c1(mrCore|clickTitle|7027) _d2(mcDragndrop|html5DragStart)" foldersn
  mailsn="7027" draggable="true" data-category="101" style>
   ▶<span class="ico_current_flag">…</span>
   ▶<ul class="mInfo">…</ul>
   ▶<div class="mTitle">…</div>
   ▶<ul class="mInfo split_cell">…</ul>
    </li>
  ▼<li class="7025_li  notRead unmark  _c1(mrCore|clickTitle|7025) _d2(mcDragndrop|html5DragStart)" foldersn
  mailsn="7025" draggable="true" data-category="213" style>
   ▶<span class="ico_current_flag">…</span>
   ▶<ul class="mInfo">…</ul>
   ▼<div class="mTitle">
    ▶<div class="name _ccr(lst.from) ">…</div>
    ▼<div class="subject ">
     ▼<a href="/read/popup/?nMailId=7025" class="_d2(mcDragndrop|html5DragStart)" draggable="true">
      ▼<span class="text _ccr(lst.title) _c1(mrCore|clickTitle|7025) _m2(mrCore|middleClickTitle|7025)"
      foldersn mailsn="7025">
       ▼<span class="folderName _ccr(lst.title) _c1(mrCore|clickTitle|7025) _m2(mrCore|middleClickTitle|7025)
       " foldersn mailsn="7025">
          <span class="blind">에 분류됨</span>
        </span>
       ▼<strong class="mail_title _ccr(lst.title) _c1(mrCore|clickTitle|7025)
       _m2(mrCore|middleClickTitle|7025)" foldersn mailsn="7025">
          <span class="blind">메일 제목:</span>
          "[현대Hmall]류기곤고객님의 적립금 10,000원이 2018/08/11 소멸 예정입니다."
```

2. 텍스트 분석

2.1. KoNLPy, NLTK

2.2. Tokenization

2.3. 불용어/연어

2.1. KoNLPy, NLTK

■ 자연어?


자연어 분석
- 형태소 분석
- 구문 분석
- 의미 분석
- 담화 분석
- 중의성 해소
뭘 할 수 있을까..?
응용 기술
- 검색
- 온라인 광고
- 자동번역
- 감정분석
- 음성인식
- 맞춤법검사

■ 자연어 분석 단계(1)

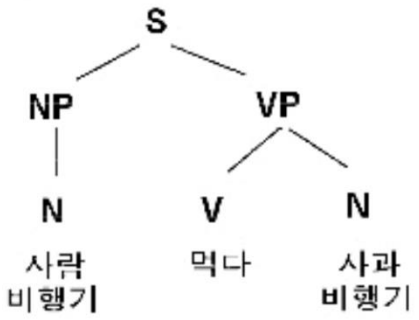
토큰 분리, 어간 추출, 품사 부착, 색인, 벡터화

문장 경계 인식, 구문분석, 공기어, 개체명 사전 구축(PLOT, 수치, 외국어 한글 표기), 개체명 인식

대용어 해소(대명사, 두문자어, 약어, 수치), 의미 중의성 해결(동명이인, 이명동인)

분류, 군집, 중복, 요약, 가중치, 순위화, 토픽 모델링, 이슈 트래킹, 평판분석, 감성분석, 복합논증분석,

■ 자연어 분석 단계(2)

형태소 분석 →	구문 분석 →	의미 분석 →	담화 분석
입력된 문장을 형태소 단위로 분할하고 품사를 부착	**주어, 목적어, 서술어와 같은 구문단위를 찾음**	**문장이 의미적으로 올바른 문장인지를 판단**	**대화 흐름상 어떤 의미를 가지는지를 찾음** - 문맥구조 분석 (문장들의 연관 관계), - 의도분석 (전후관계를 통한 실제 의도)
1) 나는 - 나+는 - 날(다)+는 - 나(다)+는 2) 과학자들에게 - 과학자 + 들 + 에게		1) 사람이 사과를 먹는다. (o) 2) 사람이 비행기를 먹는다. (x) 3) 비행기가 사과를 먹는다. (x)	1) 철수는 어항을 떨어뜨렸다. 그는 울고 말았다. 2) 철수는 우승을 했다. 그는 울고 말았다.

■ 담화

○ 문장이 연속되어 이루어지는 말의 단위, 문단

나는 급하게 옷을 주워 입었다. 개미 한 마리가 방바닥을 내 발이 있는 쪽으로 기어오고 있었다. 그 개미가 내 발을 붙잡으려고 하는 것 같은 느낌이 들어서 나는 얼른 자리를 옮겨 디디었다.

밖의 이른 아침에는 싸락눈이 내리고 있었다. 우리는 할 수 있는 한 빠른 걸음으로 여관에서 떨어져 갔다.

"난 그 사람이 죽으리라는 걸 알고 있었습니다."

안이 말했다.

"난 짐작도 못했습니다."

라고 나는 사실대로 얘기했다.

"난 짐작하고 있었습니다."

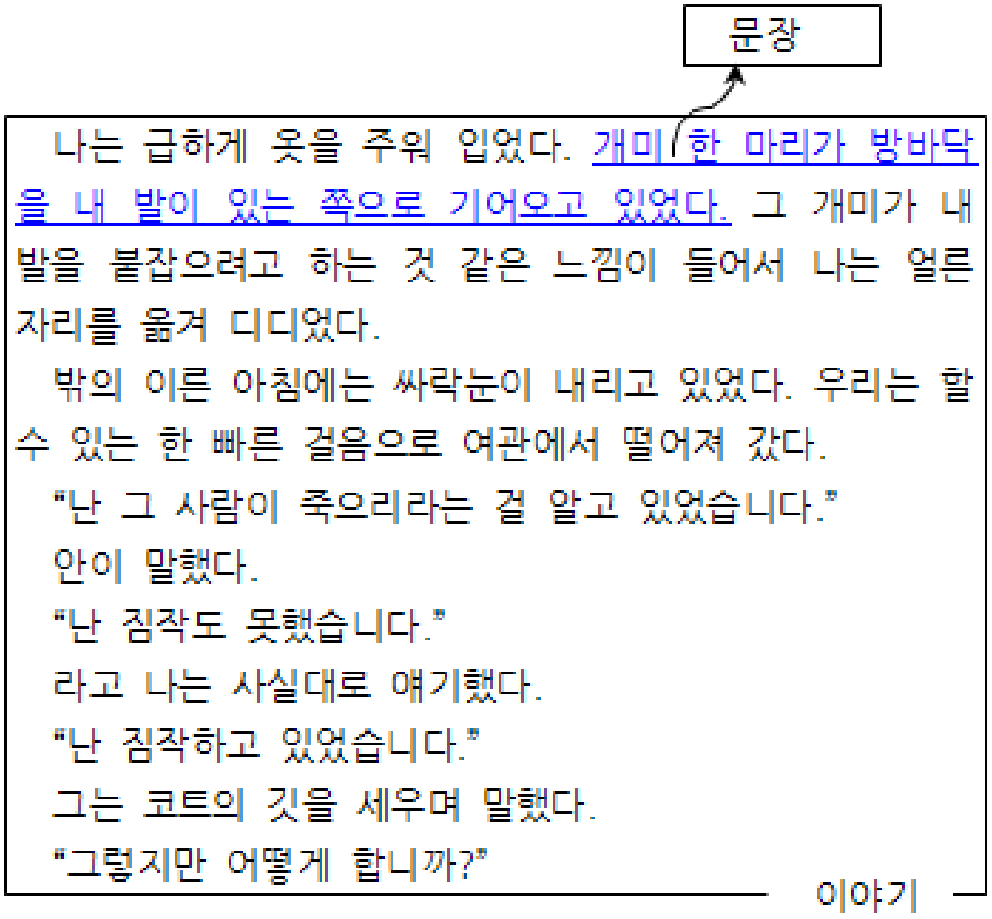
그는 코트의 깃을 세우며 말했다.

"그렇지만 어떻게 합니까?"

— 이야기

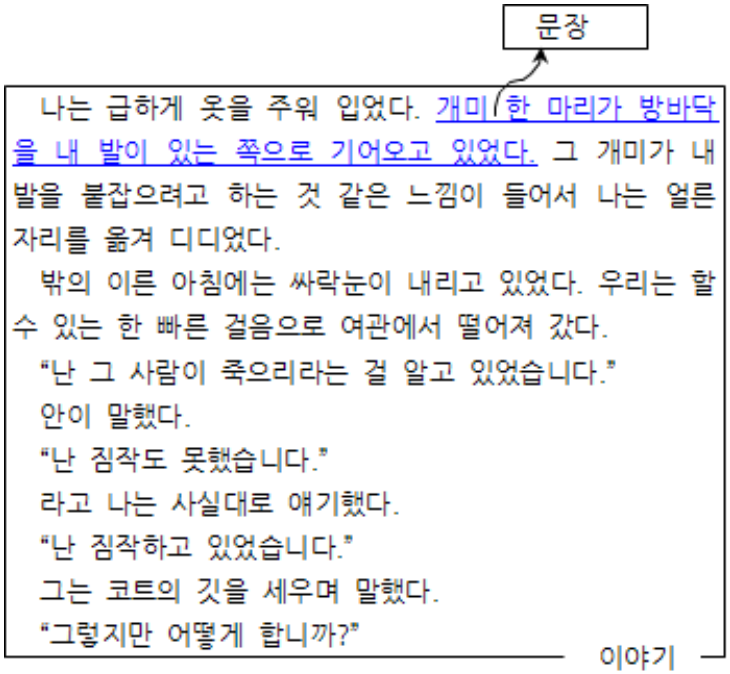
■ 문장

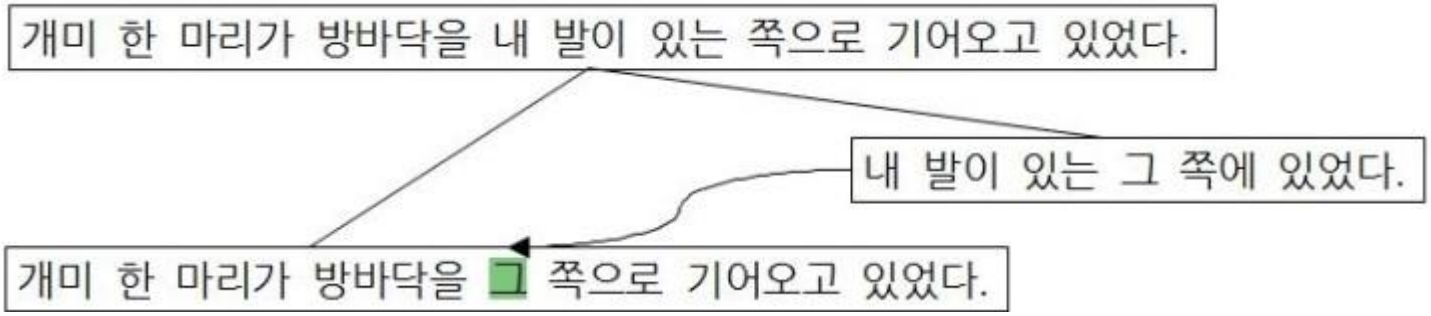
○ 완결된 내용을 나타내는 최소 단위



■ 문장

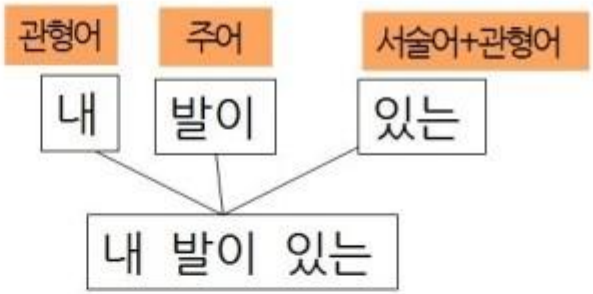
○ 완결된 내용을 나타내는 최소 단위



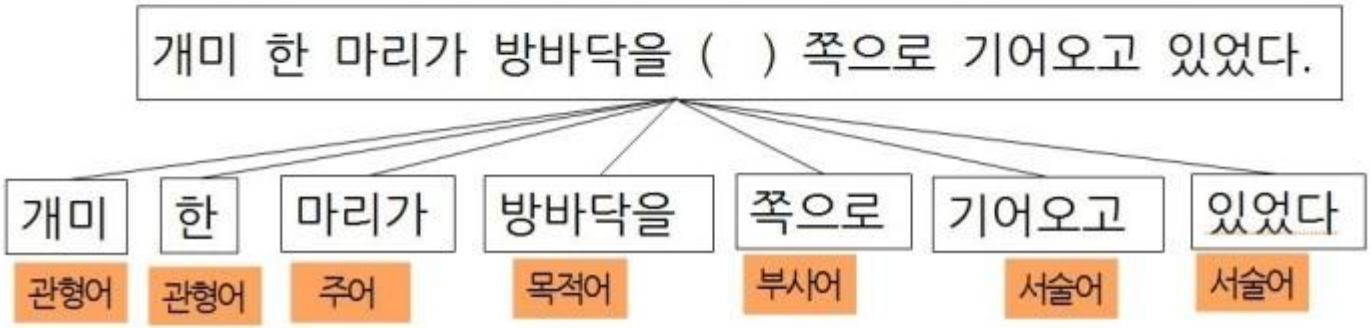


■ 어절

○ 문장을 구성하는 단위



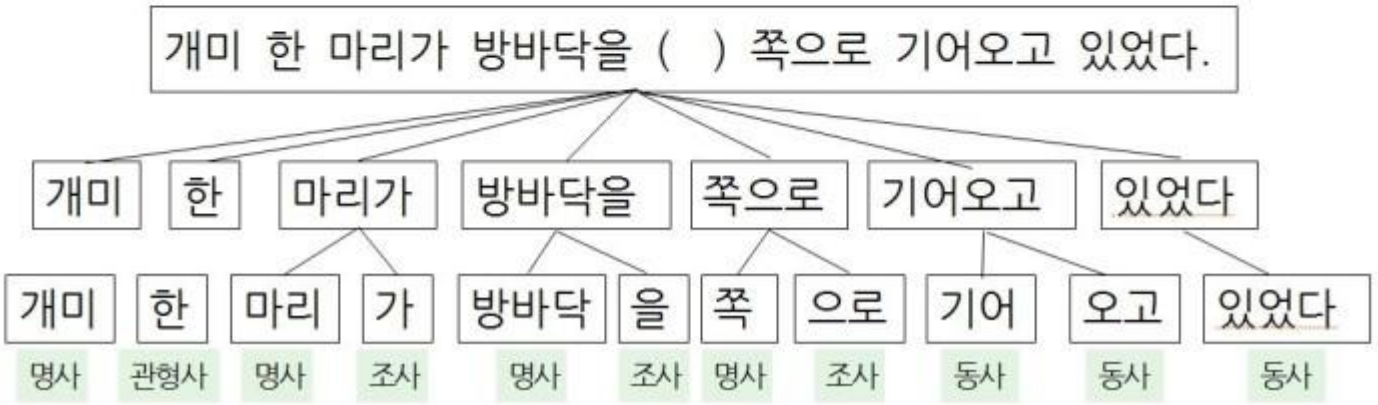
개미 한 마리가 방바닥을 내 발이 있는 쪽으로 기어오고 있었다.



■ 단어

○ 어절을 구성하는 요소

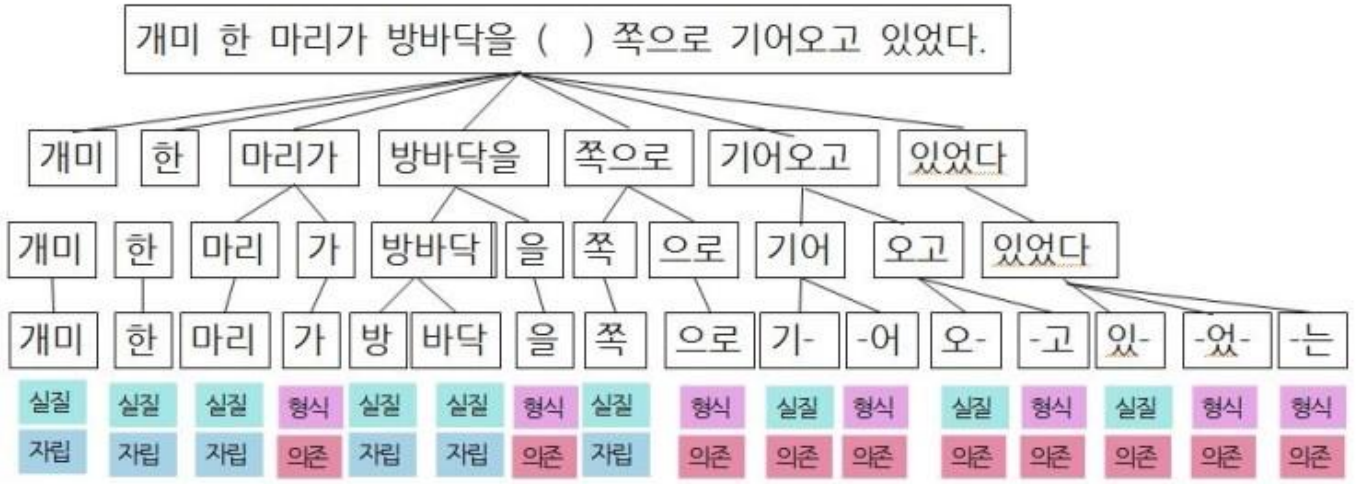
개미 한 마리가 방바닥을 내 발이 있는 쪽으로 기어오고 있었다.



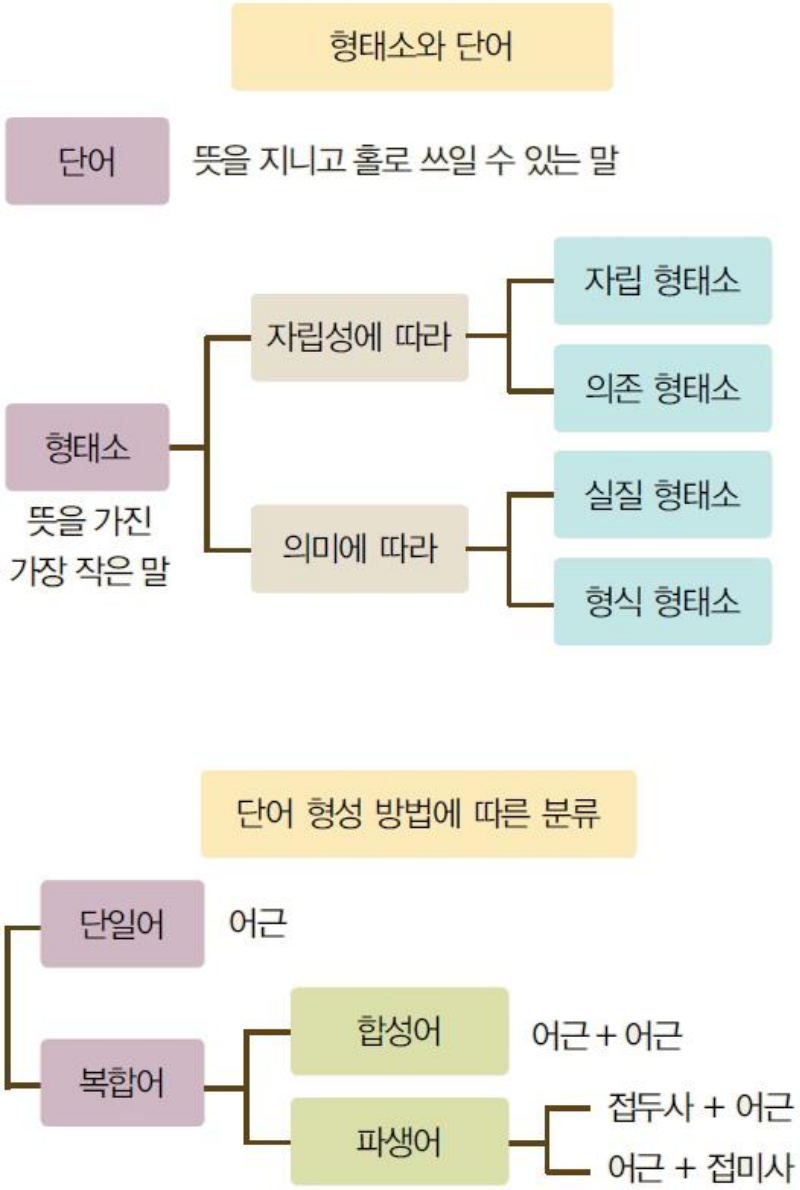
■ 형태소

○ 의미를 가진 문법의 최소 단위

개미 한 마리가 방바닥을 내 발이 있는 쪽으로 기어오고 있었다.



■ 형태소의 분류


형태소와 단어
단어
뜻을 지니고 홀로 쓰일 수 있는 말
형태소
뜻을 가진
가장 작은 말
자립성에 따라
의미에 따라
자립 형태소
의존 형태소
실질 형태소
형식 형태소
단어 형성 방법에 따른 분류
단일어
어근
복합어
합성어
어근 + 어근
파생어
접두사 + 어근
어근 + 접미사

■ KoNLPy

○ 한국어 정보처리를 위한 파이썬 패키지

○ Corpus, Morpheme Analyzer, POS Tagging, …

○ http://konlpy.org/ko/latest/

➢ Acknowledgement

– 박은정, 조성준, "KoNLPy: 쉽고 간결한 한국어 정보처리 파이썬 패키지", 제 26회 한글 및 한국어 정보처리 학술대회 논문집, 2014.

■ 설치하기

○ JDK 1.7+

○ JPype1 0.5.7+

➢ JPype1-0.5.7-cp27-none-win***(32 or _amd64)***.죄

○ KoNLPy

```
import konlpy
```

■ 설치확인 (말뭉치 불러오기)

다음의 말뭉치(corpus)를 사용할 수 있습니다:

1. `kolaw`: 한국 법률 말뭉치.
   - constitution.txt
2. `kobill`: 대한민국 국회 의안 말뭉치. 파일 ID는 의안 번호를 의미합니다.
   - 1809890.txt - 1809899.txt

```
from konlpy.corpus import kolaw

corpus = kolaw.open('constitution.txt').read()

print(corpus.split("\n")[:3])
```

■ Morpheme Analyzer

○ 한나눔(Hannanum)

➢ KAIST 시맨틱 웹 연구 센터(SWRC)에서 Java로 개발

○ 꼬마(Kkma)

➢ 서울대학교 지능 데이터 시스템 연구실(IDS Lab)에서 Java로 개발

○ 코모란(Komoran)

➢ Shineware(NLP 연구 동호회)에서 Java로 개발

○ 메카브(Mecab)

➢ 은전한닢 프로젝트에서 동경대의 일본어 형태소 분석기 Mecab를 한국어에 적용하여 개발

○ 트위터

➢ 트위터에서 오픈소스 한국어 처리기를 Scala로 개발

■ 성능 분석

○ 띄어쓰기

Hannanum	Kkma	Komoran	Mecab	Twitter
아버지가방에 들어가 / N	아버지 / NNG	아버지가방에 들어가신다 / NNP	아버지 / NNG	아버지 / Noun
이 / J	가방 / NNG		가 / JKS	가방 / Noun
시ㄴ다 / E	에 / JKM		방 / NNG	에 / Josa
	들어가 / VV		에 / JKB	들어가신 / Verb
	시 / EPH		들어가 / VV	다 / Eomi
	ㄴ다 / EFN		신다 / EP+EC	

○ 중의성

Hannanum	Kkma	Komoran	Mecab	Twitter
하늘 / N	하늘 / NNG	하늘 / NNG	하늘 / NNG	하늘 / Noun
을 / J	을 / JKO	을 / JKO	을 / JKO	을 / Josa
나 / N	날 / VV	나 / NP	나 / NP	나 / Noun
는 / J	는 / ETD	는 / JX	는 / JX	는 / Josa
자동차 / N	자동차 / NNG	자동차 / NNG	자동차 / NNG	자동차 / Noun

■ 성능 분석

○ 띄어쓰기

Hannanum	Kkma	Komoran	Mecab	Twitter
아버지가방에 들어가 / N	아버지 / NNG	아버지가방에 들어가신다 / NNP	아버지 / NNG	아버지 / Noun
이 / J	가방 / NNG		가 / JKS	가방 / Noun
시ㄴ다 / E	에 / JKM		방 / NNG	에 / Josa
	들어가 / VV		에 / JKB	들어가신 / Verb
	시 / EPH		들어가 / VV	다 / Eomi
	ㄴ다 / EFN		신다 / EP+EC	

○ 중의성

Hannanum	Kkma	Komoran	Mecab	Twitter
하늘 / N	하늘 / NNG	하늘 / NNG	하늘 / NNG	하늘 / Noun
을 / J	을 / JKO	을 / JKO	을 / JKO	을 / Josa
나 / N	날 / VV	나 / NP	나 / NP	나 / Noun
는 / J	는 / ETD	는 / JX	는 / JX	는 / Josa
자동차 / N	자동차 / NNG	자동차 / NNG	자동차 / NNG	자동차 / Noun

■ 성능 분석

○ 띄어쓰기

Hannanum	Kkma	Komoran	Mecab	Twitter
아버지가방에 들어가 / N	아버지 / NNG	아버지가방에 들어가신다 / NNP	아버지 / NNG	아버지 / Noun
이 / J	가방 / NNG		가 / JKS	가방 / Noun
시ㄴ다 / E	에 / JKM		방 / NNG	에 / Josa
	들어가 / VV		에 / JKB	들어가신 / Verb
	시 / EPH		들어가 / VV	다 / Eomi
	ㄴ다 / EFN		신다 / EP+EC	

○ 중의성

Hannanum	Kkma	Komoran	Mecab	Twitter
하늘 / N	하늘 / NNG	하늘 / NNG	하늘 / NNG	하늘 / Noun
을 / J	을 / JKO	을 / JKO	을 / JKO	을 / Josa
나 / N	날 / VV	나 / NP	나 / NP	나 / Noun
는 / J	는 / ETD	는 / JX	는 / JX	는 / Josa
자동차 / N	자동차 / NNG	자동차 / NNG	자동차 / NNG	자동차 / Noun

■ 형태소 분석

○ Hannanum

```
from konlpy.tag import Hannanum

h = Hannanum()
text = "기상청은 이번 주에도 비다운 비 소식 없이 폭염의 강도가 더 강해지고 \
        폭염 지역이 확대될 것으로 보인다며 한낮의 야외활동을 피하고 \
        실내 온도 관리에도 유의할 것을 당부했습니다."

textSplit = text.split()
candidates = h.analyze(text)

for (i, t) in enumerate(textSplit):
    print(t)

    for c in candidates[i]:
        print("   ",c)
    print("")

print(h.pos(text), end="\n\n")

print(h.morphs(text), end="\n\n")

print(h.nouns(text))
```

○ Komoran

```
from konlpy.tag import Komoran

r = Komoran()
text = "기상청은 이번 주에도 비다운 비 소식 없이 폭염의 강도가 더 강해지고 \
        폭염 지역이 확대될 것으로 보인다며 한낮의 야외활동을 피하고 \
        실내 온도 관리에도 유의할 것을 당부했습니다."

print(r.pos(text), end="\n\n")

print(r.morphs(text), end="\n\n")

print(r.nouns(text))
```

○ Twitter

```
from konlpy.tag import Twitter

t = Twitter()
text = "기상청은 이번 주에도 비다운 비 소식 없이 폭염의 강도가 더 강해지고 \
        폭염 지역이 확대될 것으로 보인다며 한낮의 야외활동을 피하고 \
        실내 온도 관리에도 유의할 것을 당부했습니다."

print(t.phrases(text), end="\n\n")

print(t.pos(text), end="\n\n")

print(t.morphs(text), end="\n\n")

print(t.nouns(text))
```

■ 말뭉치 탐색

○ Heap's law

```
from konlpy.corpus import kolaw
from konlpy.tag import Kkma as kkma
from matplotlib import pyplot as plt

k = kkma()
pos = lambda x: ['/'.join(p) for p in k.pos(x)]
docs = [kobill.open(i).read() for i in kobill.fileids()]

global_unique = []
global_unique_cnt = []
for doc in docs:
    tokens = pos(doc)
    unique = set(tokens)
    global_unique += list(unique)
    global_unique = list(set(global_unique))
    global_unique_cnt.append(len(global_unique))
    print(len(unique), len(global_unique))

# draw heap
plt.plot(global_unique_cnt)
plt.savefig('heap.png')
```

Natural Language Analysis
with Python NLTK

■ NLTK (Natural Language Toolkit)

○ Building Python programs to work with human language data

○ Provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries

■ 설치하기

○ NLTK

```
!pip install nltk
```

■ 설치확인 (말뭉치 불러오기)

```
import nltk
```

○ 말뭉치와 품사표 설치

```
nltk.download('gutenberg')
nltk.download('maxent_treebank_pos_tagger')
```

```
from nltk.corpus import gutenberg

files = gutenberg.fileids()
print(files)

corpus = gutenberg.open('austen-emma.txt').read()
print(corpus)
```

2.2. Tokenization

○ 토큰화

➢ 문장

`nltk.tokenize.sent_tokenize(text, language='english')`

Return a sentence-tokenized copy of *text*, using NLTK's recommended sentence tokenizer (currently `PunktSentenceTokenizer` for the specified language).

Parameters:

- **text** – text to split into sentences
- **language** – the model name in the Punkt corpus

```
#nltk.download('punkt')

from nltk.tokenize import sent_tokenize

sent_tokenize(corpus)
```

○ 토큰화

➢ 단어

`nltk.tokenize.word_tokenize(text, language='english', preserve_line=False)`

Return a tokenized copy of *text*, using NLTK's recommended word tokenizer (currently an improved TreebankWordTokenizer along with PunktSentenceTokenizer for the specified language).

Parameters:

- **text** (*str*) – text to split into words
- **language** (*str*) – the model name in the Punkt corpus
- **preserve_line** – An option to keep the preserve the sentence and not sentence tokenize it.

```
from nltk.tokenize import word_tokenize

for line in sent_tokenize(corpus):
    print(word_tokenize(line))
```

■ 토큰 분석

○ 어절 분리

```
pattern = corpus.split()
print(pattern)
```

```
from nltk import regexp_tokenize
pattern = r"([a-zA-Z0-9]+)+"
tokens = regexp_tokenize(corpus, pattern)
```

○ 어절 통계

```
import nltk

en = nltk.Text(tokens)
len(en), len(set(en)), en.vocab()
```

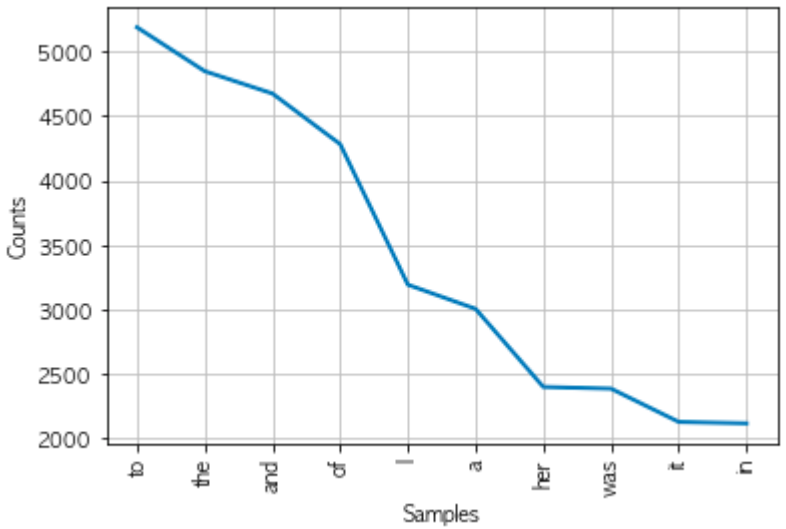
```
from konlpy.tag import Kkma
from konlpy.corpus import kolaw

kkma = Kkma()

corpus = kolaw.open('constitution.txt').read()
tokens = kkma.morphs(corpus)

ko = nltk.Text(tokens)
len(ko), len(set(ko)), ko.vocab()
```

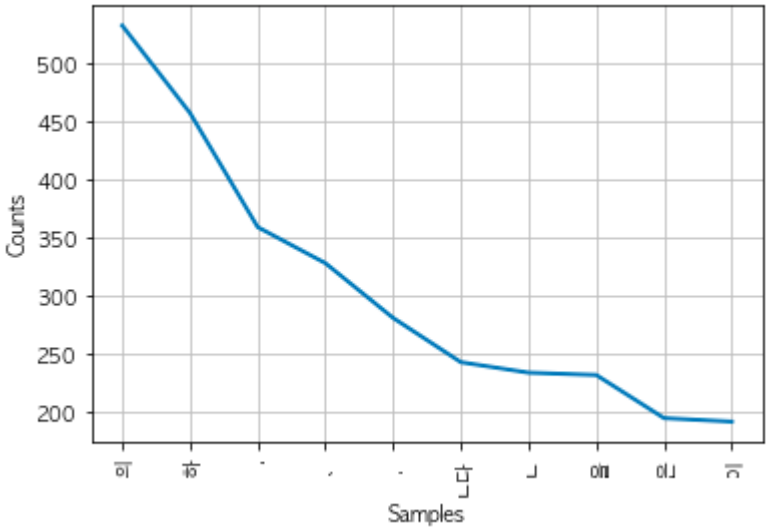
○ 어절 통계 시각화

```
print(en.plot(10))
```

➢ 한글 처리

```
from matplotlib import font_manager, rc

fontPath = "폰트 경로"
# Mac "/Library/Fonts/AppleGothic.ttf"
# Win "c:/windows/fonts/gulim.ttc"

fontFamily = font_manager.FontProperties(fname=fontPath).get_name()
rc('font', family=font_name)
```





○ 통계

```
en.count("Emma")
```

```
ko.count("대한민국")
```

○ 분포

```
en.dispersion_plot(["Emma", "Frank", "Jane"])
```

```
ko.dispersion_plot(["대한민국", "자유", "민주"])
```

○ 찾기(위치)

```
en.concordance("Emma", lines=5)
```

```
ko.concordance("대한민국", lines=5)
```

○ 유사 분포 단어 찾기

```
en.similar("Emma")

en.similar("Frank", num=5)
```

```
ko.similar("대한민국")

ko.similar("국민", num=5)
```

similar(*word*, *num=20*)

Distributional similarity: find other words which appear in the same contexts as the specified word; list most similar words first.

Parameters:

- **word** (*str*) – The word used to seed the similarity search
- **num** (*int*) – The number of words to generate (default=20)

○ n-gram 모델

```
def ngram(src, n):
    src = src.split()
    rst = []

    for i in range(len(src)):
        rst.append(src[i:i+n])

    return rst

ngram("문자열", 2)
```

```
def ngram(src, n):
    rst = []

    for i in range(len(src)):
        rst.append(src[i:i+n])

    return rst

ngram("문자열", 2)
```

○ 정규식 한글

```
from nltk import regexp_tokenize
pattern = r"([ㄱ-ㅎㅏ-ㅣ가-힣]+)+"
tokens = regexp_tokenize(corpus, pattern)
```

○ ngram 예제

```
print(ngram(tokens, 2))
```

```
for syllable in tokens:
    print(ngram(syllable, 2))
```

2.3. 정규화/연어

■ Normalization

○ 문장 부호 제거, 소(대)문자 변환, 숫자를 단어로 변환, 약어 전개 등

■ 문장 부호 제거

○ 구두점 제거

`string.punctuation`

String of ASCII characters which are considered punctuation characters in the `C` locale.

```
import re
import string
from nltk.tokenize import word_tokenize

text = ["It is a pleasant evening", "Guests, who came from US arrived at the venue", "Food was tasty."]
tokens = [word_tokenize(s) for s in text]

pattern = re.compile("[%s]" % re.escape(string.punctuation))

newTokens = []

for sentence in tokens:
    newSentence = []

    for word in sentence:
        newWord = pattern.sub("", word)

        if newWord != "":
            newSentence.append(newWord)

    newTokens.append(newSentence)

newTokens
```

■ 소문자 / 대문자 변환

```
text = "HARdwork IS KEy to SUCCESS"
print(text.lower())
print(text.upper())
```

■ Stop words

○ 문장의 전체적인 의미에 크게 기여하지 않는 단어

○ 자연어처리 기반 응용 시스템(정보 검색, 텍스트 마이닝 등)에서 탐색 공간을 줄이기 위해 사용

```
from nltk.corpus import stopwords

stopwords.fileids()

stopList = stopwords.words("english")

srcString = "Don't hesitate to ask questions."

for src in srcString.split():
    if src.lower() not in stopList:
        print(src)
```

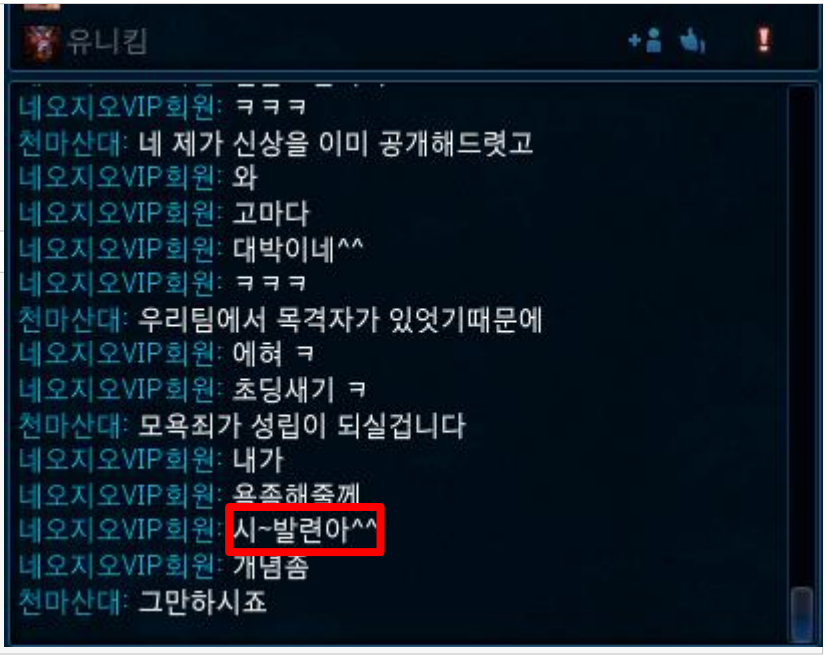
○ 한글 불용어

형태	품사	비율
이	VCP	0.01828
있	VA	0.011699
하	VV	0.009774
것	NNB	0.009733
들	XSN	0.006898
그	MM	0.005327
되	VV	0.003613
수	NNB	0.003474
이	NP	0.003361
보	VX	0.00331
않	VX	0.002976
없	VA	0.00292
나	NP	0.00269
사람	NNG	0.002074
주	VV	0.001885
아니	VCN	0.001871
등	NNB	0.001822
같	VA	0.001725
우리	NP	0.001715
때	NNG	0.001686
년	NNB	0.001648
가	VV	0.001619
한	MM	0.001584
지	VX	0.001538
대하	VV	0.001504
오	VV	0.001491
말	NNG	0.001322
일	NNG	0.00124

○ 욕설/비속어 처리

```
stopwords = ["시발", "씨발"]

srcString = "시발 련아 ^^"
for src in Kkma().morphs(srcString):
    if src not in stopwords:
        print(src)
```

```
srcString = "시발련아^^"
for src in Kkma().morphs(srcString):
    isStop = False

    for temp in ngram(src, 2):
        if temp in stopwords:
            isStop = True

    if not isStop:
        print(src)
```




```
srcString = "시~발련아^^ 시!발련아^^"
for src in srcString.split():
    src = re.sub("[^ㄱ-ㅎㅏ-ㅣ가-힣]", "", src)

    isStop = False

    for temp in ngram(src, 2):
        if temp in stopwords:
            isStop = True

    if not isStop:
        print(src)
```

■ What is Collocation?

○ Co + Location

○ 연어(Collocation)는 두 개 이상의 단어로 구성된 표현 방법

○ 특정 단어에 대한 연어는 습관적 또는 관습적으로 사용(위치)되는 방법

■ 연어의 종류

○ 명사구 / 동사구 / …

■ 품사 태깅

○ 영어

```
tokens = "The little yellow dog barked at the Persian cat".split()

enTags = nltk.pos_tag(tokens)
print(enTags)
```

○ Kkma

```
tokens = "내 친구는 잠을 많이 잔다"

koTags = Kkma().pos(tokens)
print(koTags)
```

■ 명사구 찾기

class `nltk.chunk.regexp.`**`RegexpParser`**(*grammar*, *root_label='S'*, *loop=1*, *trace=0*)

Bases: `nltk.chunk.api.ChunkParserI`

A grammar based chunk parser. `chunk.RegexpParser` uses a set of regular expression patterns to specify the behavior of the parser. The chunking of the text is encoded using a `ChunkString`, and each rule acts by modifying the chunking in the `ChunkString`. The rules are all implemented using regular expression matching and substitution.

A grammar contains one or more clauses in the following form:

```
NP:
  {<DT|JJ>}          # chunk determiners and adjectives
  }<[\.VI].*>+{      # chink any tag beginning with V, I, or .
  <.*>}{<DT>         # split a chunk at a determiner
  <DT|JJ>{}<NN.*>    # merge chunk ending with det/adj
                     # with one starting with a noun
```

```
parser_en = nltk.RegexpParser("NP: {<DT>?<JJ>*<NN.*>*}")
chunks_en = parser_en.parse(enTags)
chunks_en.draw()
```

한정사+형용사+명사류(명사, 복수 고유 등)

■ 명사구 찾기

('내', 'NP'), ('친구', 'NNG'), ('는', 'JX'), ('잠', 'NNG'), ('을', 'JKO'), ('많이', 'MAG'), ('자', 'VV'), ('ㄴ다', 'EFN')

```
parser_ko = nltk.RegexpParser("""
    NP: {<N.*>*<N.*>?}          명사류+명사류(명사, 복수 고유 등)
    VP: {<M.*>*<V.*>*<E.*>*}    부사/관형사+동사+어미
""")
chunks_ko = parser_ko.parse(koTags)
chunks_ko.draw()

for subtree in chunks_ko.subtrees():
    if subtree.label()=='NP':
        print(' '.join((e[0] for e in list(subtree))))
        print(subtree)
    elif subtree.label()=='VP':
        print(' '.join((e[0] for e in list(subtree))))
        print(subtree)

chunks_ko.draw()
```

○ 연어(collocation) 찾기

```
nltk.download("stopwords")
en.collocations()
```

```
ko.collocations()
```

collocations(*num=20*, *window_size=2*)

Print collocations derived from the text, ignoring stopwords.

Seealso: find_collocations

Parameters:

- **num** (*int*) – The maximum number of collocations to print.
- **window_size** (*int*) – The number of tokens spanned by a collocation (default=2)

■ PMI

Pointwise Mutual Information

- **Pointwise mutual information:**
  - How much more do events x and y co-occur than if they were independent?

$$\mathrm{PMI}(X,Y) = \log_2 \frac{P(x,y)}{P(x)P(y)}$$

- **PMI between two words:**
  - How much more do two words co-occur than if they were independent?

$$\mathrm{PMI}(word_1, word_2) = \log_2 \frac{P(word_1, word_2)}{P(word_1)P(word_2)}$$

○ 예제

```
from nltk import collocations

measures = collocations.BigramAssocMeasures()

print("Collocation among tagged words:")
tagged_words = Kkma().pos(corpus)
finder = collocations.BigramCollocationFinder.from_words(tagged_words)
for row in finder.nbest(measures.pmi, 5):
    print(row)

print("Collocation among words:")
words = [w for (w, t) in tagged_words]
finder = collocations.BigramCollocationFinder.from_words(words)
for row in finder.nbest(measures.pmi, 5):
    print(row)

print("Collocation among tags:")
tags = [t for (w, t) in tagged_words]
finder = collocations.BigramCollocationFinder.from_words(tags)
for row in finder.nbest(measures.pmi, 5):
    print(row)
```

■ Word Cloud

```
ko.vocab()
type(ko.vocab())

for row in ko.vocab().items():
    print(row)

import csv
with open('words.csv', 'w', encoding='utf-8') as f:
    f.write('word,freq\n')
    writer = csv.writer(f)
    writer.writerows(data)
```

■ Word Cloud

```
!pip install simplejson
!pip install pygame
!pip install pytagcloud
```

```
from konlpy.tag import Kkma
from konlpy.corpus import kolaw
from collections import Counter
import pytagcloud

kkma = Kkma()

corpus = kolaw.open("constitution.txt").read()
tokens = [k for (k,v) in kkma.pos(corpus) if v.startswith("NN")]

tags = Counter(tokens)
tags = tags.most_common(40)

tagList = pytagcloud.make_tags(tags, maxsize=40)
print(tagList)

pytagcloud.create_tag_image(tagList, "wordcloud.jpg", fontname="AppleGothic")
```

○ 한글 지원 안함, 폰트 별도 설정 필요