Data acquisition, extraction, and storage

Structured content extraction from the Web

Pierre Senellart

10 October 2025

Outline

Objective

- Assume you have crawled a somewhat homogeneous collection of Web pages with information of interest
- Given a Web page (or a collection) with a fixed structure, how to extract information of interest?
- Sometimes, extraction and crawling are interleaved (you need to know extracted content to determine what next to crawl)
- But usually good idea to crawl first, explore later (avoid having to recrawl if the extraction process changed)
- Web scraping: Entire crawl and extraction process

Wrapper

Wrapper: program (usually written in some specialized language) that extracts data from semi-structured (usually HTML) documents and turns it into a structured form (CSV, relational data, XML or JSON with fixed structure, etc.)

Wrapper induction: process of building or learning an appropriate wrapper for a collection of similarly structured documents

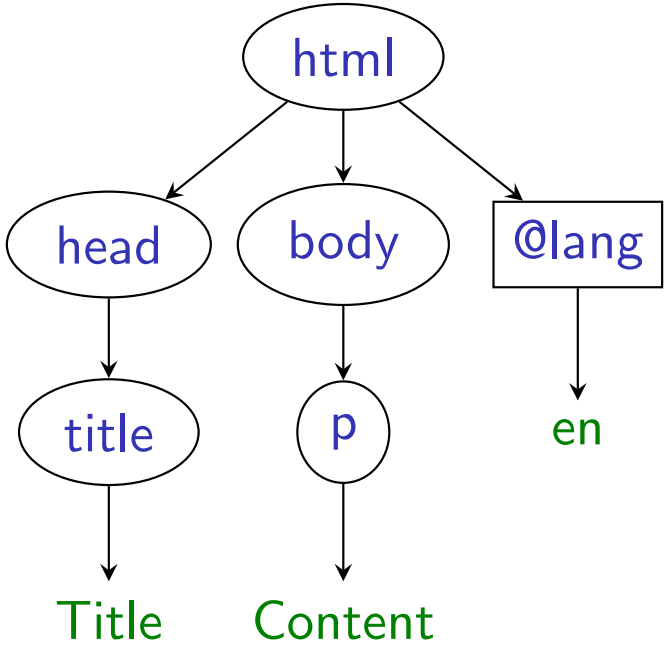
Selector: expression in a domain-specific language that is used by the wrapper to pinpoint certain regions of interest in a semi-structured document

Document Object Model (DOM)

Tree representation of an HTML document, suitable for manipulation and extraction.

Example

```
<html lang="en">
  <head><title>Title</title></head>
  <body><p>Content</p></body>
</html>
```



Languages for selectors

- Based on serialization: regular expressions (see further)
- Based on DOM:

  DOM navigation expresses local navigation in the DOM, from a node to its parent, its children, its attribute, etc. Standard API [WHATWG, 2021] but variations.

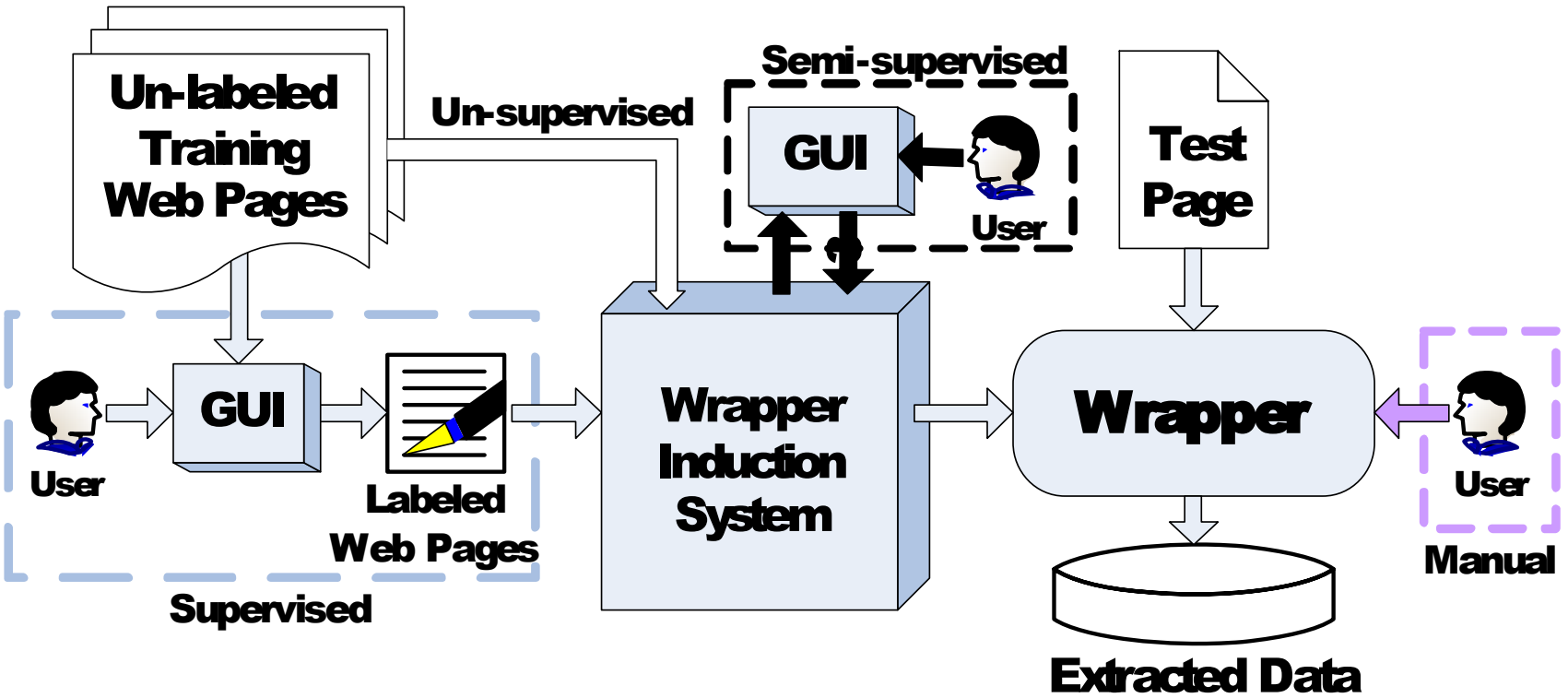
  searching elements by tag names, identifiers, names, class names

  CSS selectors (see further)

  XPath (see further)

Sometimes combined! For instance, use XPath to quickly navigate the DOM, then a regular expression to extract substrings.

Types of wrapper induction [Chang et al., 2006]



Supervised, semi-supervised, and domain-based techniques

- Many academic approaches and systems, see [Ferrara et al., 2014] for a survey
- No ready-to-use free software for supervised and semi-supervised extraction (as far as I know)
- Existing companies selling wrapper induction software
- Promising start-ups exploiting these ideas – Lixto (semi-supervised), Wrapidity (domain-based) – sold to big media intelligence groups (McKinsey, Meltwater, respectively)

Unsupervised techniques

- Exploiting data redundancy within a page [Liu et al., 2004] or across pages [Crescenzi et al., 2001, Arasu and Garcia-Molina, 2003]
- RoadRunner: one of the first such systems, freely downloadable and existing demos at `http://www.dia.uniroma3.it/db/roadRunner/`
- Usually, no labels provided by the extraction, non-trivial problem
- Nowadays: an LLM may infer for you appropriate CSS/XPath selectors, but no guarantees, and be careful of edge cases

Manual techniques

- Write a wrapper by hand, using one wrapper language (itself relying on a selector language to indicate which data to extract)
- scrapy (which also has crawling and browser simulation features) can be used to write wrapper (relying on the parsel library to support CSS and XPath selectors)
- Alternatives: BeautifulSoup in Python, regular expression or XPath libraries in arbitrary programming languages, etc.
- XSLT, a programming language for transforming XML documents, is also great for this task! `http://webdam.inria.fr/Jorge/files/wdm-pip-xslt.pdf`
- Browser instrumentors like Selenium (or Playwright, Puppeteeer) can also be made to extract data (e.g., through JS code making use of XPath expressions), since they control the browser

Outline

Regular Expressions

- Apply to the serialized representation, not to the DOM tree.
- Available in a wide range of host languages (including Python with the `re` package).
- The following characters are metacharacters.

```
? * + | ( ) ^ $ . [ ] { } " \
```

- Metacharacters have special meaning; they do not represent themselves.
- All other characters represent themselves.

Operators

r	One occurrence of r
$r?$	Zero or one occurrence of r
$r*$	Zero or more occurrences of r
$r+$	One or more occurrences of r
$r \mid s$	r or s
rs	r concatenated with s

r and s are regular expressions.

Grouping and extra symbols

- Parentheses are used for grouping.
- The expression

  ```
  ("+"|"-")?
  ```

  represents an optional plus or minus sign.
- If a regular expression begins with `^`, then it is matched only at the beginning of a line or string (depending on context).
- If a regular expression ends with $, then it is matched only at the end of a line or string (depending on context).
- The dot `.` matches any non-newline character.

Character groups

- Brackets `[ ]` match any single character listed within the brackets.
- For example,
  - `[abc]` matches `a` or `b` or `c`.
  - `[A-Za-z]` matches any letter.
- If the first character after `[` is `^`, then the brackets match any character *except* those listed.
  - `[^A-Za-z]` matches any nonletter.

Outline

CSS

- CSS: Cascading StyleSheets
- W3C recommendation
- History: CSS1 (1996), CSS2 (1998), CSS2.1 (2005), CSS3 (continuously developed and updated since 1999, decomposed into modules, each module having its own version number, named level)
- Somewhat disparate support by browsers, see `https://caniuse.com/`
- Here: focus on CSS selectors [W3C, 2022] – and in particular those supported by the cssselect Python library, itself used by parsel and thus scrapy

CSS and HTML

- HTML describes structure and content
- CSS describes: text formatting and layout of blocks with respect to one another

Simple, multiple, universal selectors

Simple selector: tag name

Multiple selector: several selectors joined by commas

Universal selector: '*', selects everything

Examples

- `ul` selects unordered lists
- `h1,h2,h3,h4,h5,h6` selects all section titles
- `*` selects everything

Class selectors

Class selector: class name, prefixed with '`.`', as it appears in a `class` attribute of an HTML tag

Examples

- `.person` selects all tags with class `person` (and possibly other classes)
- `p.comment` selects all <p> tags with class `comment`

Identifier selector

Identifier: as defined by the `id` attribute of an HTML tag. Similar to classes, but only one tag with a given `id` in the whole HTML document

Identifier selector: identifier name, prefixed with '#', as it appears in the `id` attribute of an HTML tag

Examples

- `#introduction` selects the tag with identifier `introduction`
- `p#introduction` selects the <p> tag with identifier `introduction`

Contextual selectors

Contextual selector: 2 selectors or more separated by spaces. $A\ B$ selects B's only if they are contained in A's

Child selector: 2 selectors separated by >. $A > B$ selects B's children of A's

Next sibling selector: 2 selectors separated by +. $A + B$ selects B's that are the next sibling of an A

Subsequent sibling selector: 2 selectors separated by $\sim$. $A \sim B$ selects B's that are a following sibling of an A

Examples

- `h1 em` selects text in emphasis within a main title
- `ul ol, ol ul, ul ul, ol ol` selects nested lists

Pseudo-class

Pseudo-class: specify some external properties of a class

Examples

- `article > p:first-child` selects all paragraphs that are first children of an <article>
- `article > p:nth-child(2)` selects all paragraphs that are second children of an <article>

Attribute selectors

Based on the presence or value of an attribute.

Examples

- `p[class]` select all paragraphs with a `class` attribute
- `p[class="bar"]` select all paragraphs with a `class` attribute equal to `bar` (different from `p.bar` !)
- `p[class^="bar"]` select all paragraphs with a `class` attribute starting with `bar`
- `p[class*="bar"]` select all paragraphs with a `class` attribute containing with `bar`

Outline

XPath

cf. separate set of slides

Bibliography I

Arvind Arasu and Hector Garcia-Molina. Extracting structured data from Web pages. In *SIGMOD*, pages 337–348, June 2003.

Chia-Hui Chang, Mohammed Kayed, Mohem Ramzy Girgis, and Khaled F. Shaalan. A survey of Web information extraction systems. *IEEE Transactions on Knowledge and Data Engineering*, 18(10):1411–1428, October 2006.

Valter Crescenzi, Giansalvatore Mecca, and Paolo Merialdo. RoadRunner: Towards Automatic Data Extraction from Large Web Sites. In *VLDB*, 2001.

Emilio Ferrara, Pasquale De Meo, Giacomo Fiumara, and Robert Baumgartner. Web data extraction, applications and techniques: A survey. *Knowledge-Based Systems*, 70:301–323, 2014. ISSN 0950-7051. doi: https://doi.org/10.1016/j.knosys.2014.07.007. URL `https://www.sciencedirect.com/science/article/pii/S0950705114002640`.

Bibliography II

Bing Liu, Robert L. Grossman, and Yanhong Zhai. Mining Web Pages for Data Records. *IEEE Intelligent Systems*, 19(6):49–55, 2004.

W3C. Selectors Level 4, November 2022. `https://www.w3.org/TR/2014/REC-html5-20141028/`.

WHATWG. Document Object Model. `https://dom.spec.whatwg.org/`, 2021.