

# Lecture 2: Document representation and String processing

## COMP90049 Knowledge Technologies

Sarah Erfani and Karin Verspoor, CIS

Semester 2, 2017



THE UNIVERSITY OF  
MELBOURNE

Lecture 2:  
Document  
representation  
and  
String processing

COMP90049  
Knowledge  
Technologies

Data

Data types

Doc Representation

Processing  
strategies

Pattern matching

Regular expressions

Regex

Pattern language

Pattern programming

- Data without regular, decomposable internal structure
- Examples: blogs, MP3 files, JPEG files
- In practice, most data has *some* structure to it (e.g. track titles in MP3s, document fields in PDF files)

**Lecture 2:  
Document  
representation  
and  
String processing**

COMP90049  
Knowledge  
Technologies

**Data**

**Data types**

Doc Representation  
Processing  
strategies

**Pattern matching**

Regular expressions  
Regex  
Pattern language  
Pattern programming

- Data which strictly conforms to a schema
- Consistency of data guaranteed by its origins in backend DBs
- Examples: ABN lookup, library catalogues

Lecture 2:  
Document  
representation  
and  
String processing

COMP90049  
Knowledge  
Technologies

Data

Data types

Doc Representation

Processing  
strategies

Pattern matching

Regular expressions

Regex

Pattern language

Pattern programming

- Data which conforms in part to a schema
  - irregular or incomplete data
  - data which can change in format rapidly and unpredictably
- Examples: Wikipedia entries, BibTeX records

```
@InProceedings{Gulli:Signorini:2005,  
  author = {Antonio Gulli and Alessio Signorini},  
  title = {The Indexable Web is more than 11.5 billion pages},  
  booktitle = {Proceedings of the 14th International World Wide  
  year = 2005,  
  address = {Chiba, Japan}  
}
```

# Un- or Semi- or Structured?

**Lecture 2:**  
**Document**  
**representation**  
**and**  
**String processing**

COMP90049  
Knowledge  
Technologies

**Data**

**Data types**

Doc Representation

Processing  
strategies

**Pattern matching**

Regular expressions

Regex

Pattern language

Pattern programming

- Web pages
- Excel spreadsheet
- Electronic Health Record
- Email
- Video
- Student marks database

# Text on the Web: What we see

## Lecture 2: Document representation and String processing

COMP90049  
Knowledge  
Technologies

Data

Data types

Doc Representation

Processing  
strategies

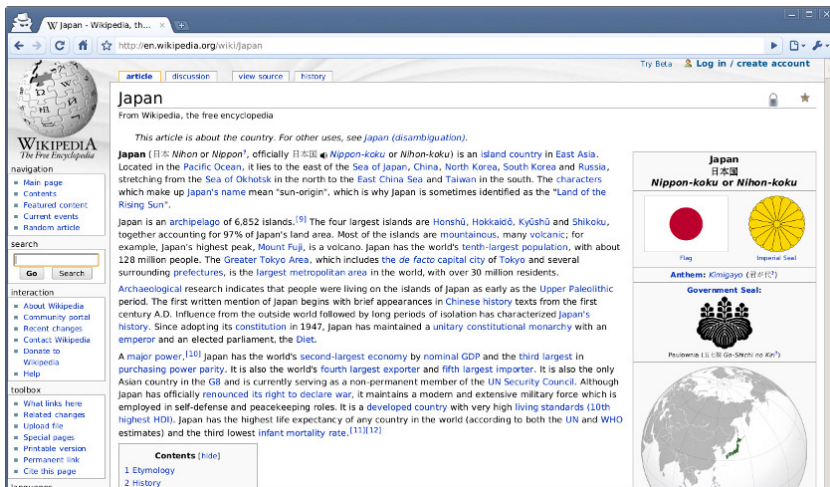
Pattern matching

Regular expressions

Regex

Pattern language

Pattern programming



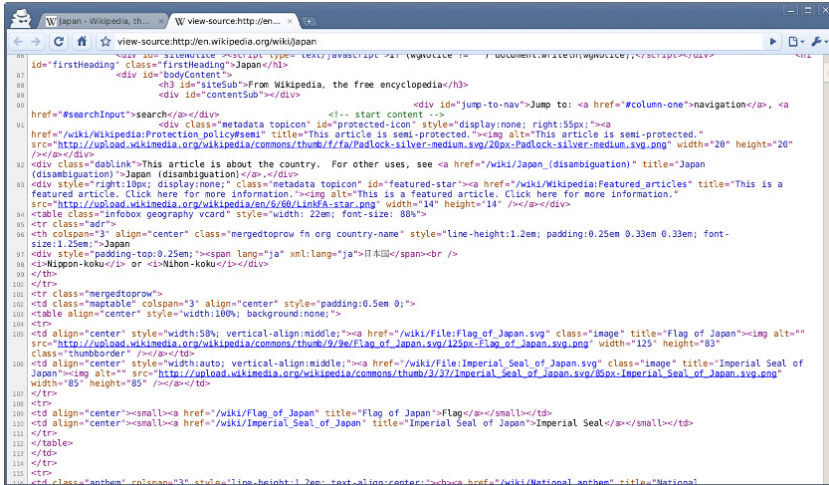
The screenshot shows the Wikipedia page for Japan. The browser address bar displays "http://en.wikipedia.org/wiki/japan". The page title is "Japan" with the subtitle "From Wikipedia, the free encyclopedia". A navigation bar includes links for "article", "discussion", "view source", and "history". The main text begins with a disambiguation note: "This article is about the country. For other uses, see *Japan (disambiguation)*." The text then describes Japan as an island country in East Asia, located in the Pacific Ocean, east of the Sea of Japan, China, North Korea, South Korea, and Russia. It mentions the Sea of Okhotsk to the north, the East China Sea and Taiwan to the south, and that the characters in its name mean "sun-origin". The text states Japan is an archipelago of 6,852 islands, with the four largest being Honshū, Hokkaidō, Kyūshū, and Shikoku, accounting for 97% of the land area. It notes Japan's highest peak is Mount Fuji, a volcano, and that Japan has the world's tenth-largest population, with about 128 million people. The Greater Tokyo Area is mentioned as the largest metropolitan area in the world, with over 30 million residents. The text discusses archaeological research indicating people have lived on the islands since the Upper Paleolithic period, with the first written mention in Chinese history texts from the first century A.D. It notes Japan's isolation since adopting its constitution in 1947, maintaining a unitary constitutional monarchy with an emperor and an elected parliament, the Diet. A major power, Japan has the world's second-largest economy by nominal GDP and the third largest in purchasing power parity. It is also the world's fourth largest exporter and fifth largest importer. It is a non-permanent member of the UN Security Council. Although Japan has officially renounced its right to declare war, it maintains a modern and extensive military force which is employed in self-defense and peacekeeping roles. It is a developed country with very high living standards (10th highest HDI). Japan has the highest life expectancy of any country in the world (according to both the UN and WHO estimates) and the third lowest infant mortality rate.

The right sidebar contains information about Japan, including the national flag (a red circle on a white field) and the imperial seal (a yellow sun with rays). It also lists the national anthem as "Kimigayo" and the government seal. At the bottom of the sidebar is a map of Japan.

The left sidebar contains navigation links for "Main page", "Contents", "Featured content", "Current events", "Random article", "Search", "Interaction", "About Wikipedia", "Community portal", "Recent changes", "Contact Wikipedia", "Donate to Wikipedia", "Help", "Toolbox", "What links here", "Related changes", "Upload file", "Special pages", "Printable version", "Permanent link", "Cite this page", and "Language".



- Data
  - Data types
- Doc Representation**
- Processing strategies
- Pattern matching**
  - Regular expressions
  - Regex
  - Pattern language
  - Pattern programming



**Lecture 2:**  
**Document**  
**representation**  
**and**  
**String processing**

COMP90049  
Knowledge  
Technologies

**Data**

Data types  
Doc Representation

**Processing**  
**strategies**

**Pattern matching**

Regular expressions  
Regex

Pattern language  
Pattern programming

- Use structure where it is available.
- Use semantics (a schema, meta-data) where it is available.
- Look for bits we 'understand'.

... But how?



# Regular expressions

## Lecture 2: Document representation and String processing

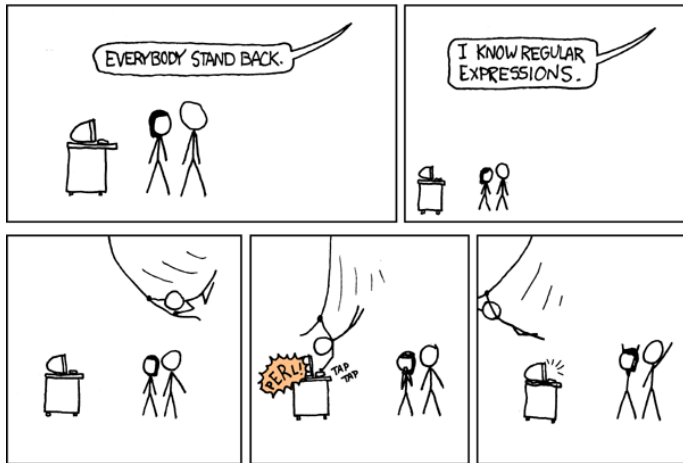
COMP90049  
Knowledge  
Technologies

### Data

Data types  
Doc Representation  
Processing  
strategies

### Pattern matching

Regular expressions  
Regex  
Pattern language  
Pattern programming



From [xkcd.com/208](http://xkcd.com/208), used under Creative Commons  
Attribution-NonCommercial 2.5 License.

Regular expressions (regex, regex) are patterns that match character strings.

They can be thought of as describing a set of strings.

- **Search:** Find the strings in a file that contain a substring that matches a given pattern (grep family).  

```
> egrep 'rudd' *.txt  
> egrep 'col(o|ou)r' *.txt
```
- **Find and replace:** Substitute some new string for the matching substring (sed, vi).  

```
s/rudd/gillard/g  
s/[dD]og/Canis lupus familiaris/g
```
- **Validate or test:** Check if new string is correct (awk, Python, Perl).  

```
$input =~ /gillard/  
$input =~ /^[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,4}$/
```

The four main concepts of regex mirror the four types of structure in imperative programming languages.

Sequence:     `i = 2; j = 3;`

Assignment:   `i = 2;`

Selection:     `if A:`  
                  `do thing`  
                  `else:`  
                  `do other thing`

Repetition:    `while True:`  
                  `i += 1`

Matching:      `/cat/`

Memoization:   `(pattern)`

Alternation:    `/cat|dog/`

Repetition:     `/(cat)*/`

As the examples above show, regular expressions are a mix of literal characters and command or control characters. For example,

- `a` means “match the character `a`”
- `|` means *or*

`{ } [ ] ( ) ^ $ . | * + ? \` are known as *metacharacters* and need to be escaped by a backslash (`\`) to be used in a literal match; for example,

`\$` means “match the character `$`”, and

`\\` means “match the character `\`”.

Beware, some tools have different metacharacters. `?` in shells means the same as `.` in standard regex.

And in some cases `\` turns a character into a metacharacter.

Here, I sometimes use `/` as a pattern delimiter. In some tools, it too is a metacharacter.

The foundation of regex is literal matching:

`/knowledge/`

- Each character matches itself.
- Matches are case sensitive.
- Whitespace is significant:  
`/over priced/` won't match "overpriced"
- Substrings are uninterpreted; they are not assumed to be whole words or have any specific semantics.  
`/lane/` will match "planet"

Another special case is newline. Many tools that incorporate regex are **line-oriented**, and either cannot match across a line break or do so in idiosyncratic ways.

Lecture 2:  
Document  
representation  
and  
String processing

COMP90049  
Knowledge  
Technologies

Data

Data types  
Doc Representation  
Processing  
strategies

Pattern matching

Regular expressions  
Regex

Pattern language

Pattern programming

The wildcard `.` is the most basic metacharacter

- Matches any single character (except a newline); good for crossword puzzles:

```
> egrep '.n.wl.d..' .../local/words.txt
    acknowledge
    acknowledged
    :
```

The anchors `^` and `$` match the start and end of a line or string, respectively.

- ```
> egrep '^n.wl.d..$' .../local/words.txt
    knowledge
```

The `|` metacharacter expresses alternation or disjunction

- `/a|b|c/` matches “a”, “b”, or “c”.
- `/cat|dog/` matches “cat” or “dog”.
- `/\$(US|AU|CD)/` matches “\$US”, “\$AU”, or “\$CD”.

A note on precedence: the `|` character has low precedence, and the parentheses in the last example are necessary.

Check – what is the difference between:

- `> grep 'ed|ing$' /usr/share/dict/words`
- `> grep '(ed|ing)$' /usr/share/dict/words`

The precise number of characters to match may be unknown; instead, we specify a repetition construction.

Some repetitions involve an arbitrary number:

- `*`: zero or more of the preceding element
- `?`: zero or one of the preceding element
- `+`: one or more of the preceding element

These are *greedy* – they match as many characters as they can. So `.*` will always match a complete string and `a.*b` will pick up the *last* “b” in the string.

Sometimes we care, but only approximately, about number.

- `{n}`: exactly  $n$  of the preceding element
- `{m,n}`: between  $m$  and  $n$  (inclusive) of the preceding element
- `{n,}`:  $n$  or more of the preceding element
- `{,m}`: up to  $m$  of the preceding element

For example, `labell?ing` matches “labeling”, “labelling”.



Sometimes, rather than one particular character or any character, we want to match any of a set of characters.

Some possible character classes:

- `/[Kk]nowledge/`
- `/[aeiou]/` –note that this is equivalent to `/a|e|i|o|u/` or `/(a|e|i|o|u)/`
- `/^\$[0-9]+/`
- `/^[A-Z][a-z]*/`
- `/[A-Za-z]+ /`

Observe that ranges can be used to denote the character classes.

Observe also that within `[,]`, metacharacters may be used in their literal meaning. For example, in some languages, the class `[\$]` matches “\” or “\$”.

Lecture 2:  
Document  
representation  
and  
String processing

COMP90049  
Knowledge  
Technologies

Data

Data types  
Doc Representation  
Processing  
strategies

Pattern matching

Regular expressions  
Regex

Pattern language

Pattern programming

A second use of the `^` metacharacter is to negate character classes.  
`/[^A-Za-z]/` matches any non-alpha character.

In some languages, `^` and `-` are the only metacharacters within ranges.  
(But see the discussion of named classes on the next slide.)

What do these match?

- `/[^0-9]/`
- `/[^"]/`
- `/<[^>>/`

Some character classes are used so frequently that they have names:

- `[0-9] = [[:digit:]] = \d`
- `[a-zA-Z0-9_] = [[:word:]] = \w`
- `[\ \t\r\n\f] = [[:space:]] = \s`

As do their negations:

- `[^0-9] = \D`
- `[^a-zA-Z0-9_] = \W`
- `[^\ \t\r\n\f] = \S`

Beware again: Which named character classes are available and how they are represented depends on the software you use.

Lecture 2:  
Document  
representation  
and  
String processing

COMP90049  
Knowledge  
Technologies

Data

Data types  
Doc Representation  
Processing  
strategies

Pattern matching

Regular expressions  
Regex  
Pattern language  
Pattern programming

Placing a pattern in parentheses leads to the match being stored as a variable.

The first stored pattern has the name `\1`, the  $n$ th is `\n`. Sadly, there is no way of operating on stored patterns, but they can be accessed for subsequent matching.

Example: What does `/([a-zA-Z]+) +\1/` match?

They are particularly powerful in string substitution.

Example: `s/([A-Z])[a-z]+ ([A-Z][a-z]+)/\1. \2/`

Now we can parse the regex from earlier on:

```
/^[A-Z0-9._%+~]+@[A-Z0-9.-]+\.[A-Z]{2,4}$
```

- `^[A-Z0-9._%+~]+`: match one or more of these characters
  - `@`: followed by an “@”
  - `[A-Z0-9.-]+`: followed by one or more of these characters
  - `\.`: followed by a dot
  - `[A-Z]{2,4}$`: followed by 2–4 upper case letters, and then end of line
- 
- What do you think this pattern is for?
  - How might this pattern be improved?

Lecture 2:  
Document  
representation  
and  
String processing

COMP90049  
Knowledge  
Technologies

Data

Data types  
Doc Representation  
Processing  
strategies

Pattern matching

Regular expressions  
Regex  
Pattern language  
Pattern programming

There are several pattern-based programming languages, in particular Python and Perl. There are also good command-line tools, in particular `sed` and `awk`. (Perl is also used in this way.)

A quick look at `awk` . . .

- Line-oriented; each block of code describes a series of operations to be applied to a line of input. Every line is processed in turn.
- Code is C-like (i.e., Java-like, C++-like).
- Lines of input are parsed into fields, and assigned to variables `$1`, `$2`, `$3`, . . .
- A line of input is only processed if it matches a pattern.
- Fields may be tested to see if they match a pattern.

```
Baughman Edward D. <Edward.Baughman@ENRON.com>
Baughman Edward <Edward.Baughman@ENRON.com>
Becker Lorraine <Lorraine.Becker@ENRON.com>
"Beck, Sally" <Sally.Beck@ENRON.com>,
Beck Sally <Sally.Beck@ENRON.com>
bejules@hotmail.com
Ben <Ben.Brasseaux@ENRON.com>
```

This is a complete awk program for processing the input above.

```
/<[ ^ ]*@ENRON[ ^ ]*>/{
    for( i=1 ; i<=NF ; i++ )
        if( $i ~ /^[A-Za-z]*$/ ) print $i;
}
```

NF is a special variable containing the number of fields in the current line. Other variables (e.g., i) are created automatically when they are referenced.

**Lecture 2:**  
**Document**  
**representation**  
**and**  
**String processing**

COMP90049  
Knowledge  
Technologies

**Data**

Data types  
Doc Representation  
Processing  
strategies

**Pattern matching**

Regular expressions  
Regex  
Pattern language  
**Pattern programming**

- What are regular expressions and what are they used for?
- What are the main concepts used in regular expressions?
- What kinds of search tasks can and cannot be addressed with regular expressions?

- Consolidate your understanding of the regular expression metacharacters; some useful references:

`docs.python.org/dev/howto/regex.html`

`perldoc perlretut` on any CIS server (or even a Mac!)

`perldoc.perl.org/perlretut.html`

`java.sun.com/docs/books/tutorial/essential/regex/`

**Next Lecture:** Similarity