Lecture 2: Document representation and String processing

Sarah Erfani and Kari

Semest

THE UNIVERSITY OF MELBOURNE

Unstructured data

s in

MP3s, document fields in PDF files)

Structured data

- Examples: ABN lookup, library catalogues

Semi-Structured data

- Data which conforms in part to a schema
  - irregular or incomplete data
  - data which can change in format rapidly and unpredictably
-

```
  author =   {Antonio Gulli and Alessio Signorini},
  title =   {The Indexable Web is more than 11.5 bil
  booktitle = {Proceedings of the 14th Int
  year =  2005,
  address = {Chiba, Japan}
}
```

Un- or Semi- or Structured?

- Video
- Student marks database

Text on the Web: What we see

Text on the Web: What the computer sees

Making sense of data

. . . But how?

Regular expressions

From `xkcd.com/208`, used under Creative Commons Attribution-NonCommercial 2.5 License.

Regular expressions

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

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

- **Find and replace:** Substitute so substring (sed, vi).

```
s/rudd/gillard/g
s/[dD]og/Canis lupus familiari
```

- **Validate or test:** Check if new string is correct (awk, Python, Perl).

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

Regular expressions

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

/cat/

(*pattern*)

/cat|dog/

```
    do thing
else:
    do other thing
```

Repetition:

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

/(cat)*/

Regular expressions

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*

acters and

\$ means "match the character $", a
\\ means "match the character \".

Beware, some tools have different metac ns 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.

Matching

The foundation of regex is literal matching:

/knowledge/

-
-
-

  /over priced/ won’t match “overpriced”
- Substrings are uninterpreted; they a 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 is idiosyncratic ways.

Matching

The wildcard . is the most basic metacharacter

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

:

The anchors ^ and $ match the start an respectively.

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

Alternation

The | metacharacter expresses alternation or disjunction

- /a|b|c/ matches “a”, “b”, or “c”.
-
-

| the

parentheses in the last example are neces

Check – what is the difference between:

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

Repetition

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
-
-

`.*`

will always match a complete string and `a.*b` will pick up the *last* “`b`” in the string.

Sometimes we care, but only approximate

- `{n}`: exactly *n* of the preceding ele
- `{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`”.

Character classes

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

Some possible character classes:

-
- or
-
- `/^[A-Z][a-z]*/`
- `/ [A-Za-z]+ /`

Observe that ranges can be used to denote th

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

Negative classes

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

nges

What do these match?

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

Named classes

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

- `[0-9] = [[:digit:]] = \d`
- `[a-zA-Z0-9_] = [[:word:]] = \w`
- [illegible]
- `[^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.

Back-references or memoization

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

, there is no

Example: What does `/([a-zA-Z]+) +`

They are particularly powerful in string subs

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

Putting it all together

Now we can parse the regex from earlier on:

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

- [illegible]
- [illegible]
- `[A-Z0-9.-]+`
- `\.`: followed by a dot
- `[A-Z]{2,4}$`: followed by 2–4 uppe [illegible] line

- What do you think this pattern is for?
- How might this pattern be improved?

Programming with patterns

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.)

-

- Code is C-like (i.e., Java-like, C++-lik

- Lines of input are parsed into fields, and $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.

Programming with patterns

```
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>
```

```
            awk
```

```
/<[^ ]*@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.

Summary

- 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

`docs.python.org/dev/howto/rege`

`perldoc perlretut` on any CIS server (

`perldoc.perl.org/perlretut.htm`

`java.sun.com/docs/books/tutori`

Next Lecture: Similarity