Regular Expressions

Problem

Sometimes we need to find specific *patterns* in a text string:

- Social Security numbers (xxx-yy-zzzz).
- URLs in a document that specify the port explicitly, rather than using the default (i.e., port 80).
- Retrieve the timestamp for each 503 error in HTTP server's logfile.
- All of the files on my desktop whose names start with "a" and that have a "doc" or "xls" extension, but not a "docx" or "xlsx" extension.

Problem (cont.)

- This is easy to describe in human terms, but bad news for computers.
- Even with a "normal" programming language, this is slow to execute and difficult to maintain: not a great combination.
- Regular expressions (regexp for friends) have been around for ages, but have a learning curve.

Regexp

- Regexp are a **pattern-matching language** that can be used
  - independently (`grep` utility in Unix).
  - within a programming language, via a separate engine.

Metacharacters

Character	Description
`.`	Matches any character except newline.
^	Start of the string
$	End of the string
\	Escape character (to match reserved characters).
[]	Character class. Matches any of a list of single-character/patterns. For instance, `[a-zA-Z]` and `[0-9]`

Matching something that repeats

- If we need all the *m* characters, we use a **complementary metacharacter** that modifies the character before it.

Complementary Metacharacters

Character	Description
`[^]`	Negation of `[]`, for instance, `[^A-Z]` means no uppercase.
`*`	Zero or more repetitions of a pattern or metacharacter.
`+`	One or more repetitions of a pattern or metacharacter.
`?`	Zero or One repetitions of a pattern or metacharacter.
`{n,m}`	Between `n` and `m` repetitions.

Examples

- `can'?t` matches "cant" and "can't".
- `ice ?cream` matches "icecream" and "ice cream".
- `U\.?S\.?A\.?` matches USA and U.S.A.

Metacharacters are greedy!

- If the regexp engine has to choose between a minimal match (without the optional character) and a maximal match (with the optional character), it will go for the maximal match.
- **Example**
  - If we search for `abc?` in the string *abcd*, the match will be on *abc*.
  - If we search for `hmm?` in the string *hmmmmmm*, the match will be on *hmm*, so just two *m* characters.

Examples

- `tera.{1,4}` matches `teradata` and `terabyte` in *teradata terabytes tera*
- `hmm+` matches all the *m* in *hmmmmmm*.
- All words in the dictionary that start with *p*: `p.`

Quick Exercise

- How many words in the dictionary contain *a* (not necessarily the first letter) followed by 3-5 characters, then *b* followed by 3-5 characters, then *c*?
- What is the last word in the dictionary that contains *ee* and also *oo*, in that order, but not necessarily adjacent?
- How many words in the dictionary contain either *aeu* or *au*, then followed by *s*?

Solutions

- `a.{3,5}b.{3,5}c`
- `ee.*oo`
- `ae?us`

Character classes

Character classes

- Let's make things more interesting: suppose I want to find all words starting with *p* and followed by a vowel.
- For this, we can use **character classes**. The regexp `p[aeiou]` will do.
- Note that the character class indicates that **one** of those characters will be the next.

Examples

- Phone numbers: 123.456.789 or 123-456-789: `[0-9]{3}[.-][0-9]{3}[.-][0-9]{3}`
- What does `[abc]+` match in *aaa abc cbbbaa adb*?

Shortcuts

Some character classes are so commonly defined that in some languages (like Python) there are shortcuts to create them:

- `\w` is the same as `[a-zA-Z0-9_]` (i.e., word character)
- `\W` is the same as `[^a-zA-Z0-9_]` (i.e., non-word character)
- `\d` is the same as `[0-9]` (i.e., numeric digit)
- `\D` is the same as `[^0-9]` (i.e., non-numeric character)
- `\s` is the same as `[ \t\r\n]` (i.e., whitespace)
- `\S` is the same as `[^ \t\r\n]` (i.e., non-whitespace)

Quick exercise

- Find all words in the dictionary that contain two consonants (i.e., non-vowels), two vowels, and then either *p* or *r*. These letters should be consecutive, and can be at the beginning, middle, or end of a word.
- Find all words that contain *e*, *f*, *g*, and/or *h* three times in a row, then any letter, and then *m*, *n*, and/or *o* twice in a row.
- Find words that contain a *q* followed by something other than *u*.

Solution

- `\w*[^aeiou]{2}[aeiou]{2}[pr]\w*`
- `[efgh]{3}\w[mno]{2}`
- `q[^u]`

Anchors

Anchors

- Anchors help us specify that a pattern should not only be matched, but be either in the start or end.
- **Examples:**
  - Words that start with *def*: `^def`.
  - Words that start with *d*, a vowel and an *f*: `^d[aeiou]f`.
  - Words that end with *exes*: `exes$`.
  - Words that end with a vowel: `[aeiou]$`.
  - Words that begin with a vowel and end with a non-vowel: `^[aeiou]\w*[^aeiou]$`.

Quick exercise

- Find words with three consecutive vowels, where the three-vowel sequence is at least 5 characters after the beginning and 5 characters before the end of the word.

Quick exercise

- Find words with three consecutive vowels, where the three-vowel sequence is at least 5 characters after the beginning and 5 characters before the end of the word.
- `^\w{5,}[aeiou]{3}\w{5,}$`

Grouping

More complex matches

- All words that end with *h*: easy, `h$`.
- All words that end with *h*, whether they are singular or plural: first attempt, `hs?$`.

More complex matches (cont.)

- **But** some words are pluralized with *es*, for instance, *watches*.
- For this, we use parenthesis for grouping characters: `h[s]?(es)?$`

Quick exercise

- Find words that start with *ex* and may end up with *ing* (optionally).

Quick exercise

- Find words that start with *ex* and may end up with *ing* (optionally).
- `^ex.*s(ing)?$`

Alternation

Motivation

- So far, our examples are only `AND` statements: we match patterns that do something AND something else AND ...
- How about `OR`?
- We can concatenate expressions with a `|` for `OR` statements.

Example/ Exercise

- Find all the `<b>` tags in an HTML document.
- Find all opening and closing `<b>` tags.
- Find all opening and closing `<b>` or `<i>` tags.
- Find all opening and closing `<b>` or `<i>` tags, *or* a header (h1, h2, h3).

Example/ Exercise

- Find all the `<b>` tags in an HTML document: `<\s*b\s*>`
- - Find all opening and closing `<b>` tags: `<\s*/?\s*b\s*>`
- Find all opening and closing `<b>` or `<i>` tags: `<\s*/?\s*[bi]\s*>`
- Find all opening and closing `<b>` or `<i>` tags, *or* a header (h1, h2, h3): `<\s*/?\s*b | h[1-3]\s*>`

Teradata Regex functions

REGEXP_SUBSTR

Extracts a substring from source string that matches a regular expression string parameter.

```
REGEXP_SUBSTR(source string,
                          regular expression string
             [,start from position number,
             nth occurrence, match argument ])
```

- `REGEXP_REPLACE` : Replace.
- `REGEXP_INSTR`

Match parameter

Value	Description
i	Matching will be case insensitive.
c	Matching will be case sensitive.
n	By default Period(.) does not match newline charater and with this parameter, period character can match the newline character as well.
m	If source string has new lines characters then '^' and '$' characters apply to each line in source string instead of the entire source string.
x	Matching will ignore whitespace.
l	Matching will return NULL instead of error if source string exceeds the maximun allowed input size.

Exercise

- On the `movies` table, extract the year into a new column.
- Remove the years from the movie titles.