Text Mining and Social Media Mining

DR. KAROLINA KULIGOWSKA & DR. HAB. JACEK LEWKOWICZ

UNIVERSITY OF WARSAW | Faculty of Economic Sciences

Regular expressions

Regular expressions, regexps

- Language with which we can match any string of characters that meets our assumptions with the help of appropriate operators (special metacharacters)
- Extremely useful in finding information in text (documents, code, log files, spreadsheets)
- Most patterns use
  - normal ASCII (letters, digits, punctuation, other symbols like %#$@!)
  - unicode characters can (any type of international text and symbols)

Regular expressions

- Letters
  - The simplest regular expression is a character
  - A, a, abc, xyz - they simply match themselves
- Example
  - abcdefg   abcde   abc
  - Pattern that matches (finds) all words from text above: abc
- Digits
  - Numbers are also just characters
  - 1, 123, 9, 0 - they simply match themselves
- Example
  - abc123xyz   123dollars   123
  - Pattern that matches (finds) all words from text above: 123

Regular expressions - metacharacters

- \d
  - any digit
  - can match any digit from 0 to 9
- . (dot)
  - any character
  - can match any single character (letter, digit, whitespace)
  - more dots mean more characters
- \. (period character)
  - in order to specifically match a period character use backslash \.
- Example
  - cat.   896.   ?=+.
  - Pattern that matches (finds) all words from text above: ...\.

Regular expressions - metacharacters

- [] (square brackets)
  - can match any single character within them
  - Inclusion: [abc] - only a, b, or c
  - Exclusion: [^abc] - not a, b, nor c
- Example
  - can   man   fan
  - Pattern that matches (finds) all words from text above: [cmf]an
- Example
  - dan   ran   pan
  - Pattern that excludes (skips) all words from text above: [^drp]an
- Example
  - hog   dog   bog
  - Pattern that matches only the live animals (hog, dog, but not bog) from text above: [^b]og

Regular expressions - metacharacters

- ^ (hat)
  - starts with
  - matches all terms that <u>begin</u> with a given pattern
- $ (dollar sign)
  - ends with
  - matches all terms that <u>end</u> with a given pattern
- ^…$ (exact match)
  - starts with and ends with
  - for the expression ^measure$, only the term *measure* will match

Regular expressions - metacharacters

- Example: we want to match the word "successful" in any text
  - ^success
  - matches only a term that begins with the word "success", for example: success, successful, successfulness
  - ful$
  - matches only a term that ends with the word „"ful", for example: successful, unsuccessful, purposeful
  - ^successful$
  - matches only the term "successful"

Regular expressions - metacharacters

- * (Kleene Star)
  - zero or more repetitions of the character that it follows (it always follows a character or group)
  - Example:
  - .*   matches zero or more of any character
  - ^m(.*)r$ matches the words: mr, monitor, mentor, mirror, etc.

- + (Kleene Plus)
  - one or more repetitions of the character that it follows (it always follows a character or group)
  - Example:
  - a+   matches one or more a's
  - ^me+   matches: me, mee, meee
  - [abc]+   matches one or more of any a, b, or c character

Regular expressions - metacharacters

- ? (question mark)
  - no occurrence or the repetition of the preceding character once (optionality)

- Example: we have a text with four lines

  1 file found?

  2 files found?

  24 files found?

  No files found.

- Task: match only the lines where one or more files were found
- Solution: use the metacharacter \d to match the number of files,
  and use the expression \d+ files? found\?   to match all the lines where files were found

Regular expressions - metacharacters

- \s
  - any whitespace
  - is extremely useful when dealing with raw input text
- Example: we have a text with four lines

```
1. abc
2.     abc
3.       abc
4.abc
```

- Task: match only the lines that have a space between the list number and 'abc'
- Solution: use the expression \d\.\s+abc to match the digit, the actual period (which must be escaped with backslash), one or more whitespace characters then the text
- Note: if we use the Kleene Star instead of the Plus, we also match the fourth line (which we actually wanted to skip)

Regular expressions - metacharacters

- | (the pipe)
  - different possible sets of characters (the alternative)
  - Example:
  - Buy more (milk|bread|juice)   matches only the strings Buy more milk, Buy more bread, Buy more juice
  - ^measure$ | ^sulfur$ matches only one of the two words: measure or sulfur

- Example: we have a text with four lines

  I love cats

  I love dogs

  I love logs

  I love cogs

- Task: match only the lines with animals (cats and dogs)
- Solution: use the expression   I love (cats|dogs)

Regular expressions - metacharacters

- Other, more advanced metacharacters

[a-z]	Characters a to z
[0-9]	Numbers 0 to 9
\w	Any Alphanumeric character
\W	Any Non-alphanumeric character
{m}	m Repetitions
{m,n}	m to n Repetitions
(…)	Match Group
(a(bc))	Match Sub-group
(.*)	Match all

Regular expressions - metacharacters

- Advanced example: we want to match the image files in any text

  - ^(IMG\d+\.png)$
  - matches the full filename

  - ^(IMG\d+)\.png$
  - only matches the part before the period character (if we only wanted to capture the filename and then match the extension)

Regular expressions - metacharacters

- Advanced example: we have a text with three lines

```
file_record_transcript.pdf
file_07241999.pdf
testfile_fake.pdf.tmp
```

- Task: match only the filenames that start with "file" and have the file extension ".pdf"

- Solution: use the expression `^(file.+)\.pdf$`

Email validation RegExp example ☺

```
(?:[a-z0-9!#$%&'*+/=?^_`{|}~-]+(?:\.[a-z0-9!#$%&'*+/=?^_`{|}~-]+)*|"(?:[\x01-
\x08\x0b\x0c\x0e-\x1f\x21\x23-\x5b\x5d-\x7f]|\\[\x01-\x09\x0b\x0c\x0e-\x7f])*")@(?:(?:[a-z0-
9](?:[a-z0-9-]*[a-z0-9])?\.)+[a-z0-9](?:[a-z0-9-]*[a-z0-9])?|\[(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-
9][0-9]?)\.){3}(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?|[a-z0-9-]*[a-z0-9]:(?:[\x01-
\x08\x0b\x0c\x0e-\x1f\x21-\x5a\x53-\x7f]|\\[\x01-\x09\x0b\x0c\x0e-\x7f])+)\])
```

- Source: http://emailregex.com

Thank you today!