Accelerating Information Technology

http://aiti.mit.edu

Ghana Summer 2012
Lecture 09 – Regular Expressions
What do the following strings have in common?

- jovana@mit.edu
- louis.a.sobel@gmail.com
- tony.blair@mail.gov.uk
- 1luvbieber@teenagegirls.hotmail.com
- xyz123@ugl.edu.gh
What do the following strings have in common?

- {some letters or numbers or dots}
- @
- {some letters or numbers and at least one dot}
Regular Expressions

• Describes patterns of text
  – No meaning associated, just characters
Examples of Regular Expressions

• "All English words that have a q without a u following"

• "Words that start and end with the same letter"

• "What text is embedded in the <H3> tag?"

• Strings that are valid email addresses.
Pattern Matching

- A Regular expression matches the string if an instance of the pattern described by the regular expression can be found in the string.
- If we say “matches in the string” may make it a little more clearer.
- Sometimes people also say that the string matches the regular expression.
Pattern Matching

• We use REs to determine if a given String matches a pattern
  – RE will return all matches to pattern in the String
  – Example:
    • Pattern = "rose"
    • String = "A Rose is a rose is a rose."
Literal Patterns

• Plain, literal text look to match exactly with parts of the text.
  – Example:
    • Pattern = "rose"
    • String = "A Rose is a rose is a rose."
  – Example:
    • Pattern "e i"
    • String = "A Rose is a rose is a rose"
Character Classes

• We can group multiple characters into character classes

• Some classes are provided by Java:
  – . → matches any single character, only stops at newline
    • Example: ".ose" matches "Rose" "rose", not "ose"
  – \s → matches whitespace
    • newline (\n), space, tab (\t)
    • Example: "\.s." matches "a b", "a\tb", not "ab"
Character Classes

• \S \rightarrow matches non-whitespace character
  – Example: "\S\S" matches "ab", "a!", not "a \\

• \d \rightarrow matches single digit

• \D \rightarrow matches single non-digit (including whitespace)

• \w \rightarrow matches word character
  – A-Z, a-z, 0-9, and ' _ ' matched
Custom Character Classes

• You can define custom character classes
  – Match true if any character in custom class matched
  – Use [ ] to denote custom character class

• Example:
  – [aeiou]: vowels
    • "a", "e" match "x" does not

• Can also specify ranges:
  – [A-Z]: uppercase letter
  – [a-z]: lowercase letter
Anchors (Position Characters)

- Anchors allow you to designate where a match can occur
  - ^ → match to beginning of String
    - Example:
      - Pattern: "^[Aa] [Rr]ose"
      - "A Rose is a rose is a rose."
  - $ → match at end of String
    - Example
      - Pattern: "rose$"
      - "A Rose is a rose is a rose"
Anchors (Position Characters)

• \b matches at word boundary:
  – Pattern "\brose" matches "rose" "rosemary", but not "primrose"
Repetition Operators

• Repetition operators allow us to denote that a (sub)pattern may repeat
  – * → zero or more repetitions
    • Example: "0*\d" matches "05" "5" "0006"
  – + → One or more repetitions
    • Example: "de+r" matches "deer" "deeer" "der" not "debr"
  – ? → exactly zero or 1 occurrence
    • Example "de[ae]?r" matches "der" "deer" "dear" not "debr" "deeer"
Grouping

• Just like math expressions you can group subpatterns using ( )
  – Pattern "(word)+" matches "word" "wordword" "wordwordwordword" not "" "wordd"
Example: Valid Email Address

• aiti@mit.edu
  – one or more word characters
  – Followed by '@'
  – Followed by word characters that has to have at least one '.' somewhere
    • Since '.' is an operator in a RE, we need to escape it
Example: Valid Email Address

\((\w)+@\w+(\.\w)+\)
Escaping

• If you want one of the RE reserved characters to appear in your pattern you must escape it:
  • \ . → literal . in pattern
  • \* → literal * in pattern
  • \{ } + ( ) are the others you must escape
Alternation

- | denotes logical OR operation
  - Think of || operator in Java

- Examples:
  - Pattern "soda|juice" matches "soda" "juice" "soda water", not "water"
  - "\w+@[\w.]*\.(net|gov|edu)"
    - Good or bad RE for emails?

- | has lowest precedence (applied last)
  - Use ( ) to avoid confusion
Examples of Regular Languages
Examples of Regular Languages

• $(0|1)^*.(0|1)^*$ - Binary floating point numbers
Examples of Regular Languages

- $(0|1)^*.(0|1)^*$ - Binary floating point numbers
Examples of Regular Languages

- $(0|1)^*.(0|1)^*$ - Binary floating point numbers
- $(00)^*$ - even-length all-zero strings
Examples of Regular Languages

• (0|1)*.(0|1)* - Binary floating point numbers

• (00)* - even-length all-zero strings
Examples of Regular Languages

• $(0|1)^*.(0|1)^*$ - Binary floating point numbers

• $(00)^*$ - even-length all-zero strings

• $1^*(01*01^*)^*$ - strings with even number of zeros
Match Strings and Regular Expressions

1. \(0(0|1)^*0\)
   a. 000000

2. \(((|0)1^*\))^*
   b. 01010
   c. 010101

3. \(((0|1)0(0|1))^*\)
   d. 101010
   e. 001100
Match Strings and Regular Expressions

1. $0(0|1)^*0$
   - a. $000000$
   - b. $01010$
   - c. $010101$
   - d. $101010$
   - e. $001100$

2. $((|0)1^*)^*$

3. $((0|1)0(0|1))^*$
Match Strings and Regular Expressions

1. $0(0|1)^*0$
   - a. $000000$
   - b. $01010$
   - c. $010101$
   - d. $101010$
   - e. $001100$

2. $((|0)1^*)^*$
   - b. $01010$
   - c. $010101$
   - d. $101010$
   - e. $001100$

3. $((0|1)0(0|1))^*$
Match Strings and Regular Expressions

1. $0(0|1)^*0$
   - a. 000000
   - b. 01010
   - c. 010101
   - d. 101010
   - e. 001100

2. $((|0)1^*)^*$

3. $((0|1)0(0|1))^*$
Match Strings and Regular Expressions

1. $0(0|1)^*0$
   - a. $000000$
   - b. $01010$
   - c. $010101$
   - d. $101010$
   - e. $001100$

2. $((|0)1^*)^*$
   - a. $000000$
   - b. $01010$
   - c. $010101$
   - d. $101010$
   - e. $001100$

3. $((0|1)0(0|1))^*$
   - a. $000000$
   - b. $01010$
   - c. $010101$
   - d. $101010$
   - e. $001100$

• All strings of 0’s and 1’s that do not contain the substring 011
Capture Groups

• ( ) also used to capture text to retrieve later
  – Latter in the RE pattern, or
  – After the matching is complete in your Java code
Capture Group used in Pattern

- All words that start and end with the same letter:
  - `\b(\w)\w*\1\b`

- \n references a capture group
  - numbered from left to right in pattern
  - \0 refers to the entire string that is matched

- All words that start and end with the same 2 letters:
  - `\b(\w)(\w)\w*\1\2\b` matches "boobo"
Named Capture Groups

• Capture groups can have names
• Easier to refer to than numbers
• “(P<first_name>) (P<last_name>)”
Greediness

• By default, repetition operators match as much text as possible.

• Example:
  – Want to match html tags.
  – Pattern "</?.*>*"
  – String: "Some <bold>Bold</bold> text"

• Fix: be more specific of what can occur in the tag:
  – Pattern: "</?[^>]*>"
More Greediness Control

• By default repetition operators try to match as much text as possible:
  – Ex pattern: "bo*o" matches "booooo"

• You can use different form of operators that are not greedy by appending ? after operator
  – Ex pattern: "bo*?o" matches "boooooo"
Matching Options

• Several options control how matching is performed:
  – These are passed to the Pattern.compile() method we will see later

• Important option:
  – (?:m): Multiline mode, ^ and $ match at newline boundaries (every line) as well as beginning and end of input
Regular Expressions in Python

- Lab! :(  

32
Questions?

I know I look complex, but I really am quite useful.
Sorry…

• Confused?
• Questions?
• How can this help you parse html?
• How can this help you parse incoming SMS messages?
• Regular Expressions can also replace text
  – Self learning!
More Resources