Regular expressions and the Corpus Query Language

Albert Gatt
Corpus search

- These notes introduce some practical tools to find patterns:
  - regular expressions
  - the corpus query language (CQL):
    - developed by the Corpora and Lexicons Group, University of Stuttgart
    - a language for building complex queries using:
      - regular expressions
      - attributes and values
In the following, regular expressions are written between forward slashes (//) to distinguish them from normal text. You do not typically need to enclose them in slashes when using them.
A regular expression is a pattern that matches some sequence in a text. It is a mixture of:
- characters or strings of text
- special characters
- groups or ranges

E.g. “match a string starting with the letter S and ending in ane”
Delimiting regexes

- Special characters for start and end:
  - `/^man/` => any sequence which begins with “man”: man, manned, manning...
  - `/man$/` => any sequence ending with “man”: doberman, policeman...
  - `/^man$/` => any sequence consisting of “man” only
Groups of characters and choices

- /[wh]ood/
  - matches wood or hood
  - [...] signifies a choice of characters

- /[^wh]ood/
  - matches mood, food, but not wood or hood
  - /[^...]/ signifies any character except what’s in the brackets
Ranges

- Some sets of characters can be expressed as ranges:
  - `/[a-z]/`
    - any alphabetic, lower-case character
  - `/[0-9]/`
    - any digit between 0 and 9
  - `/[a-zA-Z]/`
    - any alphabetic, upper- or lower-case character
Disjunction and wildcards

- `/ba./`
  - matches *bat, bad, ...*
  - `./.` means “any single alphanumeric character”

- `/gupp(y|ies)/`
  - *guppy OR guppies*
  - `/(x|y)/` means “either X or Y”
  - important to use parentheses!
Quantifiers (I)

- `/colou?r/`
  - matches *color* or *colour*

- `/govern(ment)?/`
  - matches *govern* or *government*

- `/?/` means zero or one of the preceding character or group
Quantifiers (II)

- `/ba+/`
  - matches `ba, baa, baaa...`

- `/(inkiss )+/`
  - matches `inkiss, inkiss inkiss`
  - (note the whitespace in the regex)

- `/+/` means “one or more of the preceding character or group”
Quantifiers (III)

- `/ba*/`
  - matches \( b, ba, baa, baaa \)
  - `/*/` means “zero or more of the preceding character or group”

- `/\( ba \)\{1,3\}/`
  - matches \( ba, ba ba \) or \( ba ba ba \)
  - `\{n, m\}` means “between \( n \) and \( m \) of the preceding character or group”

- `/\( ba \)\{2\}/`
  - matches \( ba ba \)
  - `\{n\}` means “exactly \( n \) of the preceding character or group”
CQL syntax (I)

- CQL queries consist of regular expressions over attributes (word, lemma or tag)
- Regex over word:
  \[\text{word}="\text{it}"\] \[\text{word}="\text{resulted}"\] \[\text{word}="\text{that}"\]
  - matches only "it resulted that"
- Regex over word with special characters:
  \[\text{word}="\text{it}"\] \[\text{word}="\text{result.}*"\] \[\text{word}="\text{that}"\]
  - matches it resulted/results that
- Regex over lemma:
  \[\text{word}="\text{it}"\] \[\text{lemma}="\text{result}"\] \[\text{word}="\text{that}"\]
  - matches any form of result (regex over lemma)
CQL Syntax II

- We can combine word, lemma and tag queries:

- Word and tag constraints:
  
  ```
  [word="it"] [lemma="result" & tag="V.*"]
  ```

  Matches only *it* followed by a morphological variant of the lemma *result* whose tag begins with V (i.e. a verb)
The empty square brackets signify “any match”.

Using complex quantifiers to match things over a span:

\[
\text{[word="confus.*" & tag="V.*"] []\{0,2\} [word="by"]}
\]

“verb beginning with *confus* tagged followed by the word *by*, with between zero and two intervening words”

- *confused by (the problem)*
- *confused John by (saying that)*
- *confused John Smith by (saying that)*
CQL summary

- A very powerful query language
  - BNC SARA client uses CQL
  - online SketchEngine uses it too
- Ideal for finding complex grammatical patterns.