Regular Expressions, Concordances

Sprachverarbeitung (VL + Ü)

Nils Reiter

April 18, 2023

INSTITUT FÜR
DIGITAL HUMANITIES
UNIVERSITÄT ZU KÖLN

Regular Expressions, Concordances

Sprachverarbeitung (VL + Ü)

Nils Reiter

April 18, 2023

INSTITUT FÜR DIGITAL HUMANITIES UNIVERSITÄT ZU KÖLN

CRETA CENTER FOR REFLECTED TEXT ANALYTICS

Introduction

- Theory: Related to finite state automatons, Chomsky hierarchy, formal grammar
- Practice: A powerful tool for manipulating large quantities of data (fast)
  - Limitations *are* based on theory – tree structures cannot be handled with REs

Introduction

- Theory: Related to finite state automatons, Chomsky hierarchy, formal grammar
- Practice: A powerful tool for manipulating large quantities of data (fast)
  - Limitations *are* based on theory – tree structures cannot be handled with REs
  - REs allow us to describe a large number of strings with a single string
  - I.e., a string to define a set of strings
  - REs are useful for: search queries, text deletion/replacement, ...

Introduction

- Theory: Related to finite state automatons, Chomsky hierarchy, formal grammar
- Practice: A powerful tool for manipulating large quantities of data (fast)
    - Limitations *are* based on theory – tree structures cannot be handled with REs
    - REs allow us to describe a large number of strings with a single string
    - I.e., a string to define a set of strings
    - REs are useful for: search queries, text deletion/replacement, ...

Example

```
... | tr -d '[[:punct:]]' | ...
```

Two Use Cases (for us)

1. To find things
   - In a text file (e.g., `poe.txt`)
2. To edit things

RE Support

- Grep supports multiple variants of REs
  - Basic regular expressions: Only for simple things
  - Extended regular expressions: Activate with `-E`
  - Perl-compatible regular expressions are the most powerful ones (activate with `-P`)
- Other environments
  - Python Library re
  - Java java.util.regex.Pattern
  - ...

RE Support

- Grep supports multiple variants of REs
  - Basic regular expressions: Only for simple things
  - Extended regular expressions: Activate with `-E`
  - Perl-compatible regular expressions are the most powerful ones (activate with `-P`)
- Other environments
  - Python Library re
  - Java java.util.regex.Pattern
  - ...

Getting Started

```
$ grep -E 'the' poe.txt
```

RE Syntax

- »Normal« characters stand for themselves (`/a/` will find the character »a«)

RE Syntax

- »Normal« characters stand for themselves (`/a/` will find the character »a«)
- Quantifiers
  - Asterisk: Repeat the previous character as many times as you want (`/a*/` will find »«, »a«, »aa«, …)
  - Plus symbol: Repeat the previous character once or more (`/a+/` will find »a«, »aa«, …)
  - Question mark: Make the previous character optional (`/a?/` will find »« and »a«)
  - Range in curly braces: Give exact numbers for the previous character (`/a{2,3}/` will find »aa« and »aaa«)

RE Syntax

- »Normal« characters stand for themselves (`/a/` will find the character »a«)
- Quantifiers
  - Asterisk: Repeat the previous character as many times as you want (`/a*/` will find »«, »a«, »aa«, …)
  - Plus symbol: Repeat the previous character once or more (`/a+/` will find »a«, »aa«, …)
  - Question mark: Make the previous character optional (`/a?/` will find »« and »a«)
  - Range in curly braces: Give exact numbers for the previous character (`/a{2,3}/` will find »aa« and »aaa«)

⚠

This is different from the command line matching of file names!

RE Syntax

Character Sets

Character groups with brackets `/[ ]/`

- `/d[ie]r/` will find »dir« and »der«
- Ranges: `/02[2-7]1/` will find »0221«, »0231«, …, but not »0201«
- Inversion: `/02[^36]1/` will find numbers except »0231« and »0261«

RE Syntax

Character Sets

Character groups with brackets `/[ ]/`

- `/d[ie]r/` will find »dir« and »der«
- Ranges: `/02[2-7]1/` will find »0221«, »0231«, …, but not »0201«
- Inversion: `/02[^36]1/` will find numbers except »0231« and »0261«

Pre-defined character classes

- `/[[:punct:]]/` Matches punctuation symbols
- `/[[:alpha:]]/` Any alphabetical character
- `/[[:space:]]/` Any whitespace character

RE Syntax

Alternatives

Alternatives can be defined with `/( | )/`

- `/(this|that)/` matches »this« and »that«
  - Can be more than two alternatives

RE Syntax

Special Characters and Symbols

- RE syntax gives many symbols special meaning
- If we want to match the symbol itself, we need to escape it with a backslash
- `/\[/` matches »[«

RE Syntax

Special Characters and Symbols

- RE syntax gives many symbols special meaning
- If we want to match the symbol itself, we need to escape it with a backslash
- `/\[/` matches »[«

Escaping

- The backslash is also used for other, non-printable characters
- `/\b/` matches a word boundary
  - Not an actual character, but a break between characters
  - Every transition from a regular character to space or punctuation
- `/\n/` matches a line break
  - (but not in grep, because grep only operates on lines)
- `/\\/` matches a back slash
- `/$/` matches the end of a line
- `/^/` matches the beginning of a line

Introduction

```
$ sed -E 's/regexp/replacement/g'
```

- sed: Stream editor
  - Text editor operating on the input stream and writing to the output stream
  - Similar to `tr` but much more powerful
- Options
  - `-E`: Extended REs
  - `s///`: Apply RE
  - Suffix `g`: Apply as often as possible, and not once per line

Introduction

```
$ sed -E 's/regexp/replacement/g'
```

- sed: Stream editor
  - Text editor operating on the input stream and writing to the output stream
  - Similar to `tr` but much more powerful
- Options
  - `-E`: Extended REs
  - `s///`: Apply RE
  - Suffix `g`: Apply as often as possible, and not once per line

Example

Replace every occurrence of Project Gutenberg by Project Nils:

```
cat poe.txt | sed -E 's/Project Gutenberg/Project Nils/g'
```

Grouping and Backreferences

- `/( )/` is used for grouping
- `\\1`, `\\2`, …, `\\9` can be used to re-insert the nth group from the regexp

Example

Replace all occurrences of »don't« with »do not«, »shouldn't« with »should not«, …

```
cat poe.txt | sed -E "s/\b([a-z]*)n't/\1 not/g"
```

Concordances

- Simple tool to inspect textual data
- Table with a search term centered, and specified context to the left and right

Concordances

- Simple tool to inspect textual data
- Table with a search term centered, and specified context to the left and right

Left context	Term	Right context
ed him of all faith in man or	woman	. He had made up his mind upon
those angels upon earth that	women	in adversity can be. It was a
, _begging _for him still. If	woman	's devotion, born with a first
⋮	⋮	⋮

Concordances

- Can we extract a concordance on the command line? Yes, we can!
- General idea
  - With `/.{20}QUERY.{20}/` and `grep -o` we can extract our query with 20 characters of context
  - But grep operates line-wise, which is a problem if query is near the end or beginning of a line
  - We thus need everything on a single line:
    - Insert a space before each line end, using `sed`
    - Remove all line breaks with `tr -d` (\n, \r, \f, to be on the safe side)
    - Unify all space to be a single space with `sed`
    - Feed the output into `grep -o`

demo

Section 3

Exercise

Exercise

Let's extract a concordance (from poe or any other text)!

- Insert a space before each line end
- Remove all line breaks
- Unify all space to be a single space
- Feed the output into `grep -o` and inspect the concordance
- Our query includes the context in characters. Can you extend it such that we get tokens?

Query Ideas

- How does Poe write about men and women, how about cats and dogs?
- How did he use colors, e.g. red and green? What are things that are red, which things are green?
- Poe is a known horror author. Does he use the word »fear« as a noun or verb? In which contexts?