Text::Perfide::BookCleaner,
a Perl module to clean and normalize plain text books

André Santos, José João Almeida

Abstract

Several tasks which include text documents processing, such as text alignment, format conversions or information retrieval are heavily affected by the state of the original documents.

`Text::Perfide::BookCleaner` is a Perl module to pre-process plain text books and clean them, getting them prepared for further arbitrary use.

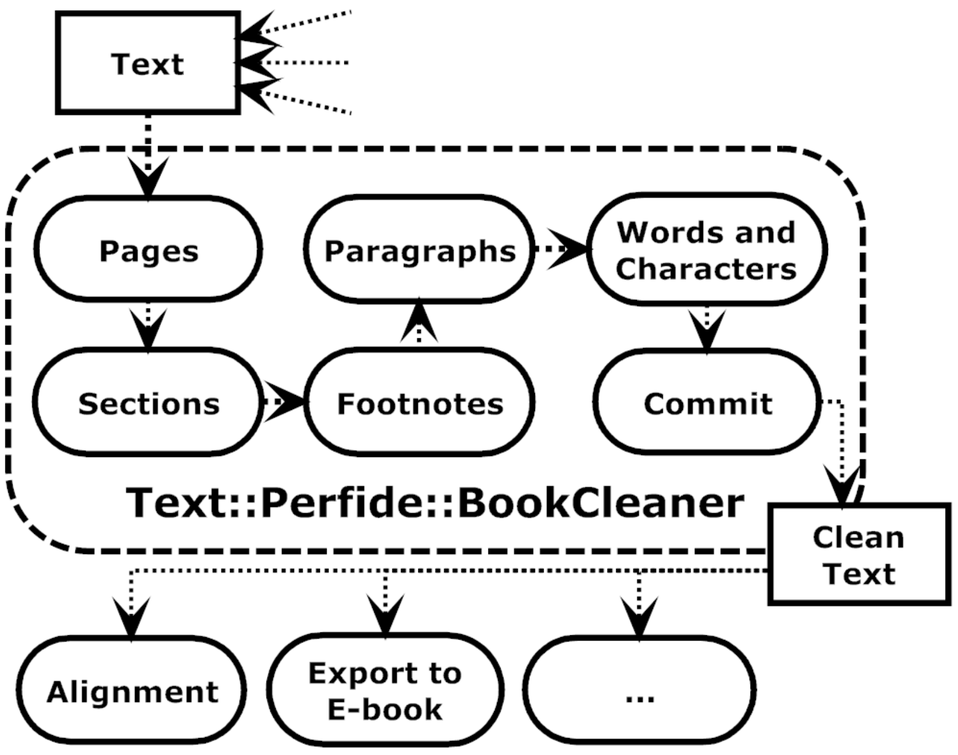
Cleaning tasks include removing page breaks, page numbers, headers and footers; finding section titles and boundaries; removing footnotes and normalizing paragraph notation and Unicode characters. The process is guided with the help of declarative objects such as ontologies and configuration files.

1 Motivation

Text processing (and particularly book alignment) are tasks whose performance is often negatively affected by:

- Page structure
- Section titles and boundaries
- Footnotes
- Paragraphs
- Character encoding etc.

2 Book cleaner



2.1 Pages

Structural elements left from a document's previous page division: page breaks, page numbers, headers and footers.

Algorithm

```
Function pages(text){
  page breaks = { page break characters ∪ page numbers }
  for ( x ∈ { page breaks }){
    prec ← lines preceding x  ; footers = footers ∪ { prec }
    foll ← lines following x  ; headers = headers ∪ { foll }
  }
  Ignore headers/footers with number of occurrences < threshold
  for ( y ∈ { headers ∪ footers }){
    Replace headers/footers with custom mark
    Add headers/footers to standoff file
  }
}
```

Table 1: Example excerpt before and after processed with **pages**

```
le nabab, et assez audacieux
pour s'emparer de sa personne.

      Page 3
^LLa maison à vapeur Jules Verne

  Le faquir, - évidemment le seul
entre tous que ne surexcitât pas
```

⇒

```
le nabab, et assez audacieux
pour s'emparer de sa personne. _pb2_
  Le faquir, - évidemment le seul
entre tous que ne surexcitât pas
```

2.2 Sections

Delimiting sections allows to:

- automatically generate tables of contents;
- identify sections from one version of a book that are missing on another version;
- matching and comparing the sections of two books before aligning them (**alignability**);
- perform additional processing on specific sections.

Algorithm

```
Function sections(text,ontology){
  lines = lines from text
  for ( x ∈ lines: x ∈ ontology ){
    Add mark to text (before x)
  }
}
```

Table 2: Example excerpt before and after processed with **sections**

```
PRIMEIRA PARTE

FANTINE


^LLIVRO PRIMEIRO

UM JUSTO

O abade Myriel

Em 1815, era bispo de Digne, o
reverendo Carlos Francisco Bemvindo
```

⇒

```
_sec+N:part=1_ PRIMEIRA PARTE

FANTINE

_sec+N:book=1_ LIVRO PRIMEIRO

UM JUSTO

O abade Myriel

Em 1815, era bispo de Digne, o
reverendo Carlos Francisco Bemvindo
```

2.3 Paragraphs

Several possible notations for paragraphs: new-line separated, blank-line separated, indented or not. Direct speech can also be delimited with quotation marks or preceded with a dash.

Some metrics based on the number of words, white spaces, empty and non-empty lines can be used to infer the type of notation used, which can then be normalized.

Algorithm

```
Function paragraphs(text){
  Calculate text's word and
    line metrics
  Find paragraph notation
  Normalize paragraph notation
}
```

2.4 Footnotes

Footnotes are formed by a call mark in the middle of the text, and the matching footnote expansion, which generally appears at the end of the same page.

Footnote expansions can often be found at the bottom of the page or, sometimes, at the end of the document.

Footnote call marks appear in the middle of the text, usually consisting of some kind of numeration highlighted by some special symbol.

Algorithm

```
Function footnotes(text)
  lines = lines from text
  fnpattern =  <<int>>, [int], ^int, ...
  for ( x ∈ dom(lines)
        : x begins with fnpattern
        ∧ x is immediately before or
            after page break ){
    Replace x in text with custom mark
    Add x to standoff file
  }
  for ( y ∈ text : y ∈ fnpattern ){
    Replace y in text with custom mark
    Add y to standoff file
  }
}
```

Table 3: Example excerpt before and after processed with **footnotes**

```
auprès de la rue Saint-Antoine,
à la porte des Tournelles[1].

[1] La Bastille, qui fut prise par le
peuple de Paris, le 14 juillet 1789,
puis démolie. B.

^LQuel était en chemin l'étonnement
 de l'Ingénu! je vous le laisse à
```

⇒

```
auprès de la rue Saint-Antoine, à
la porte des Tournelles_fnr29_.
_fne8_
^LQuel était en chemin l'étonnement
de l'Ingénu! je vous le laisse à
```

2.5 Words and characters

Sanitizing elements such as:

- Unicode characters
- Text encoding
- Glyphs
- Translineation
- Transpagination

2.6 Commit

Final step, which irreversibly removes all marks left behind by the previous steps and outputs the clean version of the text.

3 Sections Ontology

An ontology is used to store information relevant to the sectioning process: types of sections and their relations, common section names, and ordinal and cardinal numbers. Each element in the ontology is translated to several different languages.

Table 4: Excerpts from the **sections ontology**.

```
cap
PT capítulo, cap,
   capitulo
FR chapitre, chap
EN chapter, chap
RU глава
NT sec
```

```
cena
PT cena
FR scène
EN scene
BT act
```

```
end
PT fim
FR fin
EN the_end
BT _alone
```

```
3
PT terceiro,
   terceira,
   três, tres
EN third, three
FR troisième,
   troisieme
BT _numeral
```

- ontology is being used to directly create the Perl code containing the complex data structures which are used in the process of sectioning;
- the code is more readable and easier to maintain;
- the ontology can be used to discuss details with people with no programming experience.