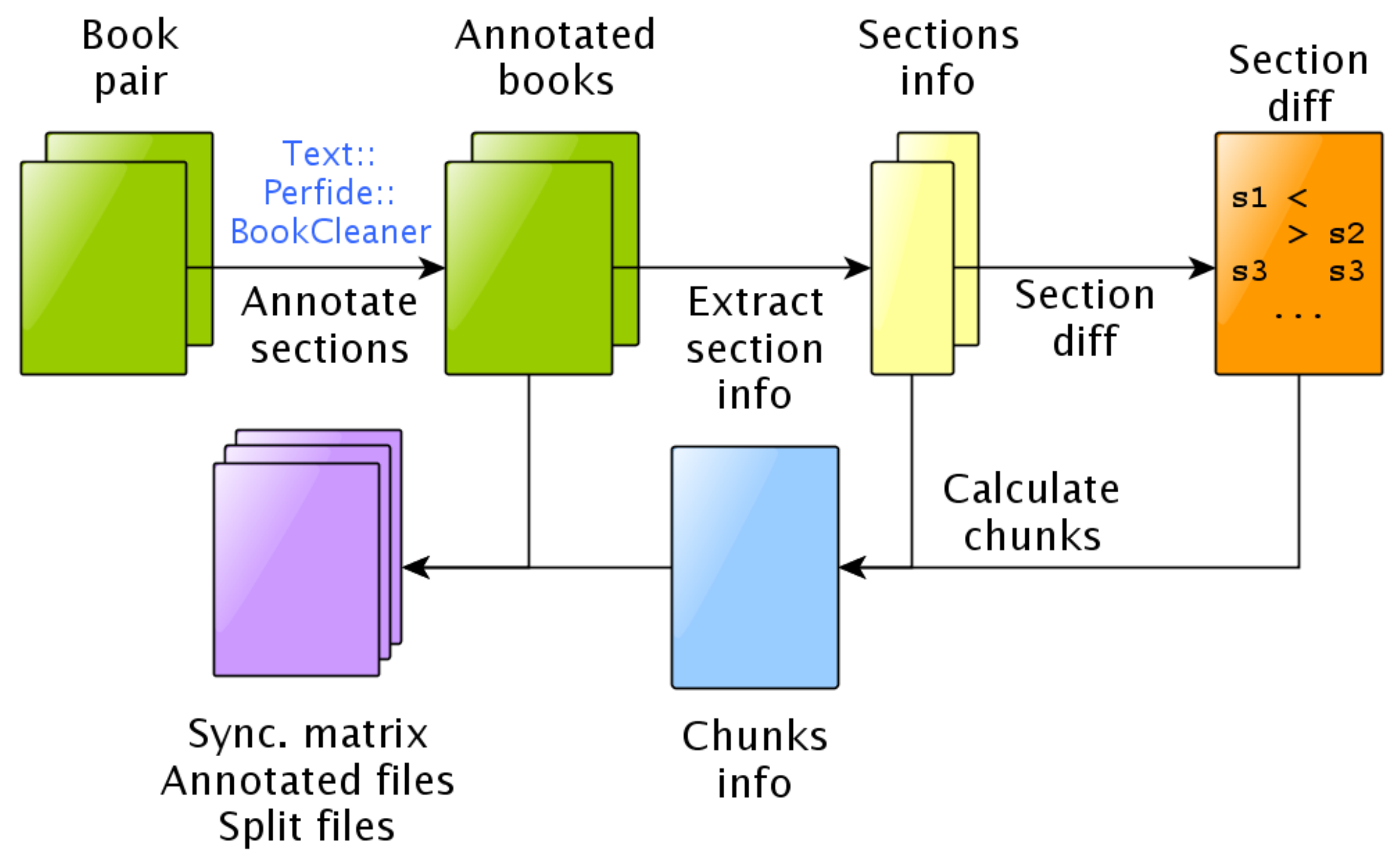
Structural alignment of plain text books

André Santos, José João Almeida, Nuno Carvalho

Abstract

Text alignment is one of the main processes for obtaining parallel corpora. When aligning two versions of a book, results are often affected by unpaired sections – sections which only exist (or could only be found) in one of the versions of the book. We developed `Text::Perfide::BookSync`*, a Perl library which performs* ***book synchronization*** *(structural alignment based on section delimitation).*

`Text::Perfide::PartialAlign` *is an extension of the partialAlign.py tool bundled with* **`hunalign`** *which proposes an alternative methods for splitting bitexts.*



1 Book synchronization

A common problem which one deals with when aligning literary works is the existence of *unmatched sections*: entire sections which exist in one version of the book and do not have a match in another version.

Text::Perfide::BookSync uses section headings to synchronize books – align them at section level, helping in the creation of anchor points which can be used to guide the aligner.

1.1 Extracting section information

Section boundaries:	Another Perl library, Text::Perfide::BookCleaner, is used to annotate section headings. These annotations are later used to determine section boundaries.
Short ID:	The section type and number are used to create a short ID which will later be used to compare sections.
Title and initial words:	These are extracted to provide users with intuitive ways of understanding the results of the synchronization.
Section size:	The number of words of two sections can be used to assess their compatibility in terms of size.

1.2 Synchronization method

The section alignment is performed as follows:

1. A short ID is generated for each section, containing its type and number (if any).
2. Short IDs from all the sections in each book are printed to a file.
3. Files are compared using Unix's *diff* command.
4. *diff*'s output shows which sections can be paired and which ones are unpaired.

1.3 Ghost sections and chunks

- Often, sections not found in one version are not actually missing – they were simply not identified.
- These sections cannot be synchronized because they are *invisible* to the synchronizer.
- Solution is to create chunks: a **chunk** is a data structure which includes a pair of matching sections, and all the following unpaired sections from both documents until the next pair of matching sections
- Every matched pair of sections will be at the beginning of a chunk, and every unpaired section will be in a chunk with a matching section at the top.
- Synchronization is then the alignment of chunks based on their first section.

```
Function ChunksCalc($_{pairs,\, secs_{L1},\, secs_{L2}}$) $: chunk^*$
Input: pairs: list of matching sections,
secs$_{L1}$: list of sections from text$_{L1}$,
secs$_{L2}$: list of sections from text$_{L2}$
Output: chunks: list of (section*, section*)
begin
    $c \leftarrow new\ Chunk$
    push($chunks, c$)
    while $secs_{L1} \neq \emptyset \wedge secs_{L2} \neq \emptyset$ do
        $s_{L1} \leftarrow$ next($secs_{L1}$)
        while $s_{L1} \notin pairs$ do
            push($c_{L1}$,$s_{L1}$)
            $s_{L1} \leftarrow$ next($secs_{L1}$)
        $s_{L2} \leftarrow$ next($secs_{L2}$)
        while $s_{L2} \notin pairs$ do
            push($c_{L2}$,$s_{L2}$)
            $s_{L2} \leftarrow$ next($secs_{L2}$)
        push($chunks, c$)
```

2 Output objects

2.1 Annotated files

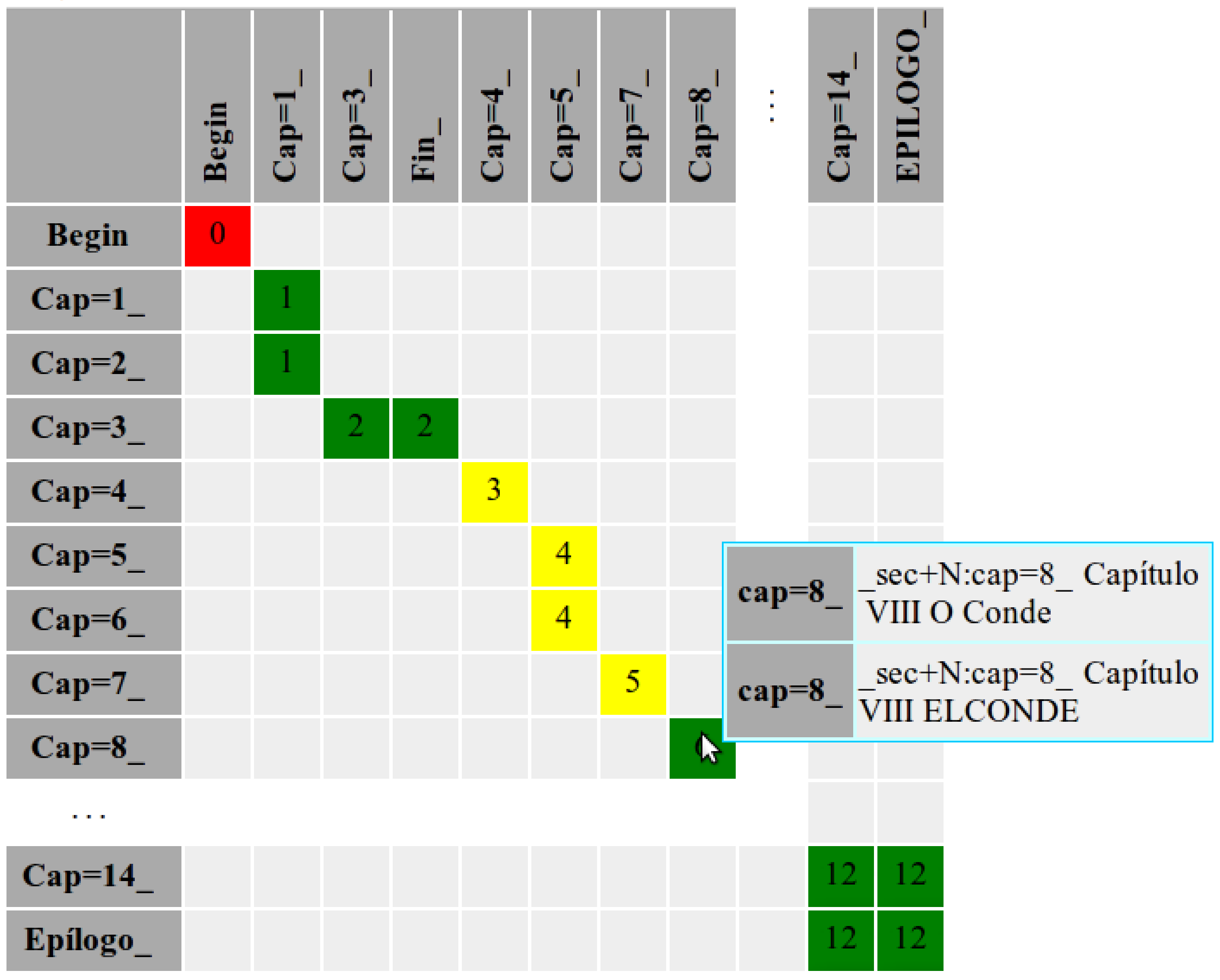
```
alice_EN
<sync id="0">
ALICE'S ADVENTURES IN
WONDERLAND
Lewis Carroll (...)
<sync id="1">
CHAPTER I. Down the
Rabbit-Hole
(...)
<sync id="2">
CHAPTER II. The Pool
of Tears
(...)
```

```
alice_ES
<sync id="0">
Las Aventuras de Alicia
en el País de las Mara-
villas, por Lewis Carrol
(...)
<sync id="1">
Capítulo 1 - EN LA MADRI-
GUERA DEL CONEJO (...)
<sync id="2">
Capítulo 2 - EL CHARCO
DE LAGRIMAS
(...)
```

2.2 Split files

```
alice_EN.txt
alice_ES.txt

     ⇓

alice_EN.txt.c000
alice_ES.txt.c000

alice_EN.txt.c001
alice_ES.txt.c001

alice_EN.txt.c002
alice_ES.txt.c002
(...)
```

2.3 Synchronization matrix

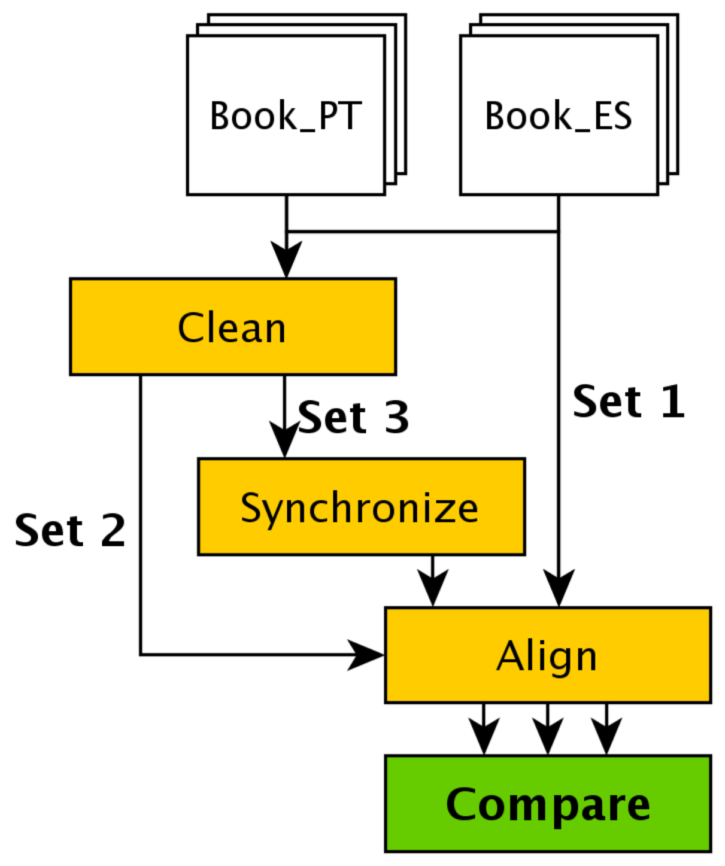


	Begin	Cap=1_	Cap=3_	Fin_	Cap=4_	Cap=5_	Cap=7_	Cap=8_	⋮	Cap=14_	EPILOGO_
Begin	0										
Cap=1_		1									
Cap=2_		1									
Cap=3_			2	2							
Cap=4_					3						
Cap=5_						4					
Cap=6_						4					
Cap=7_							5				
Cap=8_											
…											
Cap=14_										12	12
Epílogo_										12	12

cap=8_	_sec+N:cap=8_ Capítulo VIII O Conde
cap=8_	_sec+N:cap=8_ Capítulo VIII ELCONDE

3 Evaluation

- set of 20 pairs of books (Portuguese and Spanish versions)
- 3 copies of the set:
  - **Set 1:** aligned normally
  - **Set 2:** cleaned (with **bookcleaner**) and aligned
  - **Set 3:** cleaned, synchronized (with **booksync**) and aligned
- compare alignment results



	Set 1	Set 2	Set 3	$\Delta\%_{S1,S3}$
Total aligned	38	40	40	+5.0%
Classified as bad	9	8	3	-66.7%
Percentage bad	23	20	7.5	
Not aligned	2	0	0	-100%

Classified as bad: The aligner classifies as bad any alignment with more than 30% non-1:1 correspondences.

Not aligned: This happens when the aligner unexpectedly quits while processing a bitext (for example, because it ran out of memory).

4 Partial alignment

hunalign uses an auxiliary Python script, `partialAlign.py`, to split large bitexts in pairs of smaller files before alignment, using terms which **occur only once in each half of a bitext**. Text::Perfide::PartialAlign is a Perl library which implements the same approach and extends it to allow the use of **UCTS**.

```
Function T::P::PartialAlign($text_{L1}, text_{L2}, l\_ucts$) : $partial\_doc^*$
Input: text$_{L1}$: text in language $L1$, text$_{L2}$: text in
        language $L2$, l_ucts: UCTS*
Output: partial_docs: smaller files containing parts
        of the input pair.
UCTS: (word*,word*)
unique_pairs: (word,word)*
$bow = bag\_of\_words(text_{L1}, text_{L2})$
forall the $word \in dom(bow)$ do
    $ucts \leftarrow$search($l\_ucts, word$)
    if $\exists!\ w1 \in ucts_{L1} : occurs(w1, text_{L1}) = 1$
    then
        if $\exists!\ w2 \in ucts_{L2} : occurs(w2, text_{L2}) = 1$
        then push($unique\_pairs, (w1, w2)$)
$chain = extract\_longest\_chain(unique\_pairs)$
$partial\_docs = split(text_{L1}, text_{L2}, unique\_pairs)$
```

UCTS: *Unambiguous-concept translation set.* Words/terms that have a small amount of ambiguity, and are expected to be translated always the same way.

$$\left\{ \begin{array}{c} \text{wolphram} \\ \text{tungsten} \end{array} \right\}_{en} \Leftrightarrow \left\{ \begin{array}{c} \text{volfrâmio} \\ \text{tungsténio} \end{array} \right\}_{pt}$$

$$\left\{ \text{Israel} \right\}_{pt} \Leftrightarrow \left\{ \begin{array}{c} \text{Израиль} \\ \text{Израилем} \\ \text{Израиля} \\ \text{Израилю} \end{array} \right\}_{ru}$$

CPAN: `Text::Perfide::BookSync`, `Text::Perfide::PartialAlign`, `HTML::Auto`. Contact information: `andrefs@cpan.org`, `jj@di.uminho.pt`, `narcarvalho@di.uminho.pt`, `http://natura.di.uminho.pt`.