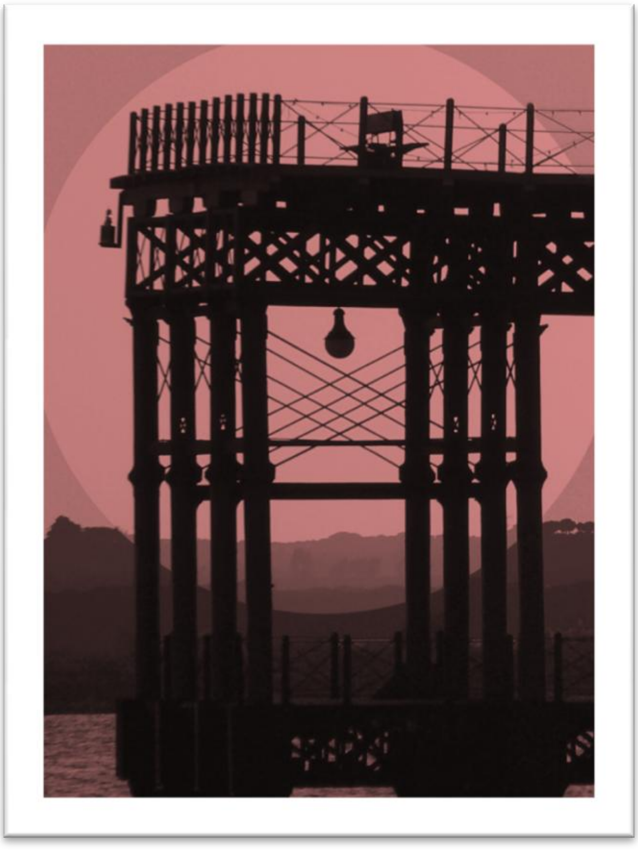
Actas del XXVII Congreso de la Sociedad Española para el Procesamiento del Lenguaje Natural

Huelva

5 - 7 Septiembre de 2011



Comité de Organización

Presidente:

Manuel J. Maña López

Miembros:

Mariano A. Crespo Azcárate
Noa P. Cruz Díaz
Manuel de la Villa Cordero
Juan Luis Domínguez Olmedo
Jacinto Mata Vázquez
Victoria Pachón Álvarez
Miguel Ángel Vélez Vélez

Comité Científico

Presidente:

Manuel J. Maña López	Universidad de Huelva

Miembros:

José Gabriel Amores	Universidad de Sevilla
Toni Badía	Universitat Pompeu Fabra
Manuel de Buenaga	Universidad Europea de Madrid
Sylviane Cardey-Greenfield	Centre de Recherche Lucien Tesnière. Besançon, France
Irene Castellón	Universitat de Barcelona
Arantza Díaz de Ilarraza	Euskal Herriko Unibertsitatea
Manuel de la Villa Cordero	Universidad de Huelva
Juan Luis Domínguez Olmedo	Universidad de Huelva
Antonio Ferrández	Universitat d'Alacant
Florentino Fernández Riverola	Universidade de Vigo
Mikel Forcada	Universitat d'Alacant
Ana García-Serrano	UNED
Alexander Gelbukh	Instituto Politécnico Nacional, México
Koldo Gojenola	Euskal Herriko Unibertsitatea
Xavier Gómez Guinovart	Universidade de Vigo
José María Gómez Hidalgo	Optenet
Julio Gonzalo	UNED
Ramón López-Cózar Delgado	Universidad de Granada
José Miguel Goñi	Universidad Politécnica de Madrid
Bernardo Magnini	Fondazione Bruno Kessler, Italia
Nuno J. Mamede	Instituto de Eng. de Sistemas e Computadores em Lisboa, Portugal
José Mariño	Universitat Politècnica de Catalunya
M. Antonia Martí	Universitat de Barcelona
María Teresa Martín	Universidad de Jaén
Patricio Martínez-Barco	Universitat d'Alacant
Raquel Martínez	UNED
Paloma Martínez Fernández	Universidad Carlos III de Madrid
Jacinto Mata Vázquez	Universidad de Huelva
Ruslan Mitkov	Universidad de Wolverhampton
Manuel Montes y Gómez	Instituto Nacional de Astrofísica, Óptica y Electrónica, México
Roser Morante	University of Antwerp, Bélgica

Lidia Moreno	Universitat Politècnica de València
Lluís Padró	Universitat Politècnica de Catalunya
Victoria Pachón Álvarez	Universidad de Huelva
Manuel Palomar	Universitat d'Alacant
Ferrán Pla	Universitat Politècnica de València
German Rigau	Euskal Herriko Unibertsitatea
Horacio Rodríguez	Universitat Politècnica de Catalunya
Leonel Ruiz Miyares	Centro de Lingüística Aplicada de Santiago de Cuba
Emilio Sanchís	Universitat Politècnica de València
Kepa Sarasola	Euskal Herriko Unibertsitatea
Isabel Segura Bedmar	Universidad Carlos III de Madrid
Mariona Taulé	Universitat de Barcelona
L. Alfonso Ureña	Universidad de Jaén
Felisa Verdejo Maillo	UNED
Manuel Vilares Ferro	Universidad de A Coruña
Luis Villaseñor-Pineda	Instituto Nacional de Astrofísica, Óptica y Electrónica, México

Revisores adicionales

Emmanuel Aguiano Hernández	Instituto Nacional de Astrofísica, Óptica y Electrónica, México
Alexandra Balahur	Universitat d'Alacant
Zoraida Callejas	Universidad de Granada
Arantza Casillas	Euskal Herriko Unibertsitatea
Alberto Díaz	Universidad Complutense de Madrid
Manuel-Carlos Díaz Galiano	Universidad de Jaén
Miguel A. García Cumbreras	Universidad de Jaén
Meritxell González	Universitat Politècnica de Catalunya
David Griol	Universidad Carlos III de Madrid
Antonio Juárez-González	Instituto Nacional de Astrofísica, Óptica y Electrónica, México
Gorka Labaka Intxauspe	Euskal Herriko Unibertsitatea
Marina Lloberes	Universitat de Barcelona
Héctor Llorens	Universitat d'Alacant
Elena Lloret	Universitat d'Alacant
Montserrat Maritxalar Anglada	Euskal Herriko Unibertsitatea
José Luis Martínez-Fernández	Universidad Carlos III de Madrid
Joaquim Moré	Universitat Oberta de Catalunya
Antoni Oliver González	Universitat Oberta de Catalunya
Jesús Peral	Universitat d'Alacant
José Manuel Perea Ortega	Universidad de Jaén
Juan A. Pérez Ortiz	Universitat d'Alacant
Felipe Sánchez Martínez	Universitat d'Alacant
Esaú Villatoro Tello	Instituto Nacional de Astrofísica, Óptica y Electrónica, México

Preámbulo

Anualmente la Sociedad Española para el Procesamiento del Lenguaje Natural (SEPLN) organiza un Congreso que pretende ser un foro de publicación de artículos científico-técnicos inéditos de calidad relevante en el ámbito del Procesamiento de Lenguaje Natural (PLN) tanto para la comunidad científica nacional e internacional, como para las empresas del sector. El XXVII Congreso de la SEPLN ha sido organizado por el Laboratorio de Recuperación de Información y Minería de Textos y Datos de la Universidad de Huelva y tendrá lugar los días 5 a 7 de septiembre de 2011. El Congreso contará con la presentación de trabajos que incluirán artículos originales, presentaciones de proyectos en marcha y descripciones de herramientas. Así mismo, se llevarán a cabo dos talleres asociados: *ICL: Workshop on Iberian Cross-Language NLP tasks* y *DDIExtraction2011: First Challenge Task: Drug-Drug Interaction Extraction*.

Las áreas temáticas tratadas en el Congreso fueron las siguientes:

- Modelos lingüísticos, matemáticos y psicolingüísticos del lenguaje.
- Lingüística de corpus.
- Desarrollo de recursos y herramientas lingüísticas.
- Gramáticas y formalismos para el análisis morfológico y sintáctico.
- Semántica, pragmática y discurso.
- Lexicografía y terminología computacional
- Resolución de la ambigüedad léxica.
- Aprendizaje automático en PLN.
- Generación textual monolingüe y multilingüe.
- Traducción automática.
- Reconocimiento y síntesis del habla.
- Extracción y recuperación de información monolingüe, multilingüe y multimodal.
- Sistemas de búsqueda de respuestas.
- Análisis automático del contenido textual.
- Resumen automático.
- PLN para la generación de recursos educativos.
- PLN para lenguas con recursos limitados.
- Aplicaciones industriales del PLN.
- Sistemas de diálogo.
- Análisis de sentimientos y opiniones.
- Minería de texto.
- Evaluación de sistemas de PLN.
- Implicación textual y paráfrasis.

Todos los trabajos presentados en el XXVII Congreso de la SEPLN han sido aceptados mediante el proceso de revisión tradicional y ha sido llevado a cabo según el calendario previsto. Queremos agradecer a los miembros del Comité Científico y a los revisores adicionales la labor que han realizado.

En esta edición, se recibieron 78 trabajos, de los cuales 60 eran artículos científicos y 18 correspondían a resúmenes de proyectos de investigación y descripciones de herramientas. De entre los 60 artículos recibidos, 33 fueron finalmente seleccionados para su presentación oral, lo cual fija una tasa de aceptación del 55%. Adicionalmente, se aceptaron otros 12 artículos para su exposición en forma de póster. Autores de otros 10 países han participado en los trabajos presentados en esta edición. Estos países son: Polonia, Bulgaria, Singapur, Portugal, Noruega, Cuba, México, EEUU, Argentina y Brasil.

El Comité Científico del Congreso se ha hecho cargo de la revisión de los trabajos. Este proceso de revisión es de doble anonimato, se mantiene oculta la identidad de los autores que son evaluados y de los revisores que realizan las evaluaciones. En un primer paso cada artículo ha sido examinado de manera ciega o anónima por tres revisores. En un segundo paso, para aquellos artículos que tenían una divergencia mínima de tres puntos (sobre siete) en sus puntuaciones sus tres revisores han reconsiderado su evaluación en conjunto. Finalmente, la evaluación de aquellos artículos que estaban en posición muy cercana a la frontera de aceptación ha sido supervisada por más miembros del Comité.

Estimamos que la calidad de los artículos es alta. El criterio de corte adoptado ha sido la media de las tres calificaciones, siempre y cuando haya sido igual o superior a 5 sobre 7.

Julio de 2011
El Comité de Organización

Pósteres

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

Text::Perfide::BookCleaner, un módulo Perl para limpieza de libros em texto plano

André Santos
Departamento de Informática
Universidade do Minho
Braga, Portugal
`pg15973@alunos.uminho.pt`

José João Almeida
Departamento de Informática
Universidade do Minho
Braga, Portugal
`jj@di.uminho.pt`

Resumen: En este trabajo se presenta `Text::Perfide::BookCleaner`, un módulo Perl para pre-procesamiento de libros en formato de texto plano para posterior alineamiento u otros usos. Tareas de limpieza incluyen: eliminación de saltos de página, números de página, encabezados y pies de página; encontrar títulos y fronteras de las secciones; eliminación de las notas de pie de página y normalización de la notación de párrafo y caracteres Unicode. El proceso es guiado con la ayuda de objetos declarativos como ontologías y ficheros de configuración. Se realizó una evaluación comparativa de las alineaciones con y sin `Text::Perfide::BookCleaner` sobre la cual se presentan los resultados y conclusiones.
Palabras clave: corpora paralelos, alineamiento de documentos, herramientas de PLN

Abstract: This paper presents `Text::Perfide::BookCleaner`, an application to pre-process plain text books and clean them for any further arbitrary use, e.g., text alignment, format conversion or information retrieval. 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. A comparative evaluation of alignments with and without `Text::Perfide::BookCleaner` was performed, and the results and conclusions are presented.
Keywords: parallel corpora, document alignment, NLP tools

1 *Introduction*

In many tasks related to natural language processing, text is the starting point, and the quality of results that can be obtained depends strongly on the text quality itself.

Many kinds of *noise* may be present in texts, resulting in a wrong interpretation of:

- individual characters,
- words, phrases, sentences,
- paragraph frontiers,
- pages, headers, footers and footnotes,
- titles and section frontiers.

The variety of existing texts is huge, and many of the problems are related to specific types of text.

In this document, the central concern is the task of cleaning text books, with a special focus on making them ready for alignment.

1.1 Context

The work presented in this paper has been developed within Project Per-Fide, a project which aims to build a large parallel corpora (Araújo et al., 2010). This process involves gathering, preparing, aligning and making available for query thousands of documents in several languages.

The documents gathered come from a wide range of sources, and are obtained in several different formats and conditions. Some of these documents present specific issues which prevent them to be successfully aligned.

1.2 Motivation

The alignment of text documents is a process which depends heavily on the format and conditions of the documents to be aligned (Véronis, 2000). When processing documents of literary type (i.e. books), there are specific issues which give origin to additional difficulties.

The main characteristics of the documents that may negatively affect book alignment are:

File format: Documents available in PDF or Microsoft Word formats (or, generally, any structured or unstructured format other than plain text) need to be converted to plain text before being processed, using tools such as `pdftotext` (Noonburg, 2001) or Apache Tika (Gupta and Ahmed, 2007; Mattmann and Zitting, 2011). This conversion often leads to loss of information (text structure, text formatting, images, etc) and introduction of noise (bad conversion of mathematical expressions, diacritics, tables and other) (Robinson, 2001);

Text encoding format: There are many different available text encoding formats. For example, Western-European languages can be encoded as ISO-8859-1 (also known as Latin1), Windows CP 1252, UTF-8 or others. Discovering the encoding format used in a given document is frequently not a straightforward task, and Dealing with a text while assuming the wrong encoding may have a negative impact on the results;

Structural residues: Some texts, despite being in plain text format, still contain structural elements from their previous format (for example, it is common to find page numbers, headers and footers in the middle of the text of documents which have been converted from PDF to plain text);

Unpaired sections: Sometimes one of the documents to be aligned contains one or more sections which are not present in the other document. It is a quite common occurrence with forewords to a given edition, preambles, etc.

Sectioning notation: There are endless ways to represent and numerate the several divisions and subdivisions of a document (parts, chapters, sections, etc). Several of these notations are language-dependent (e.g. Part One or *Première Part*). As such, aligning section titles is not a trivial task. Identifying these marks can not only help to prevent this, but it can even be used to guide the alignment process by pairing the sections first (*structural alignment*).

These problems are often enough to derail the alignment process, producing unacceptable results, and are found frequently enough to justify the development of a tool to preprocess the books, cleaning and preparing them for further processing.

There are other tools available which also address the issue of preparing corpora for alignment, such as the one described by (Okita, 2009); or text cleaning, such as TextSoap, a program for cleaning text in several formats (UnmarkedSoftware, 2011).

2 Book Cleaner

In order to solve the problems described in the previous section, the first approach was to implement a Perl script which, with the help of UNIX utilities like `grep` and some regular expressions, attempted to find the undesired elements and normalize, replace or delete them. As we started to have a better grasp on the task, it became clear that such a naive approach was not enough.

It was then decided to develop a Perl module, `Text::Perfide::BookCleaner` (`T::P::BC`), divided in separate functions, each one dealing with a specific problem.

2.1 Design Goals

`T::P::BC` was developed with several design goals in mind. Each component of the module meets the following three requirements:

Optional: depending on the conditions of the books to be cleaned, some steps of this process may not be desired or necessary. Thus, each step may be performed independently from the others, or not be performed at all;

Chainable: the functions can be used in sequence, with the output of one function passing as input to the next, with no intermediary steps required;

Reversible: as much as possible, no information is lost (everything removed from the original document is kept in separate files). This means that, at any time, it is possible to revert the book to a previous (or even the original) state.

Additionally, after the cleaning process, a diagnostic report is generated, aimed at helping the user to understand the problems detected in the book, and what was performed in order to solve them.

Below is presented an example of the diagnostic produced after cleaning a version of *La maison à vapeur* from Jules Verne:

```
footers = ['( Page _NUM_ ) = 240'];
headers=$VAR1 =
 ["(La maison \x{e0} vapeur Jules Verne) = 241"];
ctrL=1;
pagnum_ctrL=241;
sectionsO=2;
sectionsN=30;
word_per_emptyline=3107.86842105263;
emptylines=37;
word_per_line=11.5658603466849;
lines=10211;
To_be_indented=1;
words=118099;
word_per_indent=52.2793271359008;
lines_w_pont=3263;
```

In this diagnostic, we can see, among other things, that this document had headers composed by the book title and author name, footers contained the word `Page` followed by a number (presumably the page number) and that the 241 pages were separated with ^L.

The process is also configurable with the help of an ontology, and the use of configuration files written in domain-specific languages is also planned. Both are described in section 2.3.

2.2 Pipeline

We decided to tackle the problem in five different steps: *pages*, *sections*, *paragraphs*, *footnotes*, and *words and characters*. Each of these stages deals with a specific type of problems commonly found in plain text books. A *commit* option is also available, a final step which irreversibly removes all the markup added along the cleaning process. The different steps of this process are implemented as functions in `T::P::BC`.

Figure 1 presents the pipeline of the `T::P::BC` module.

`T::P::BC` accepts as input documents in plain text format, with UTF-8 text encoding (other encoding formats are also supported). The final output is a summary report, and the cleaned version of the original document, which may then be used in several processes, such as alignment (Varga et al., 2005; Evert, 2001), ebook generation (Lee, Guttenberg, and McCrary, 2002), or others.

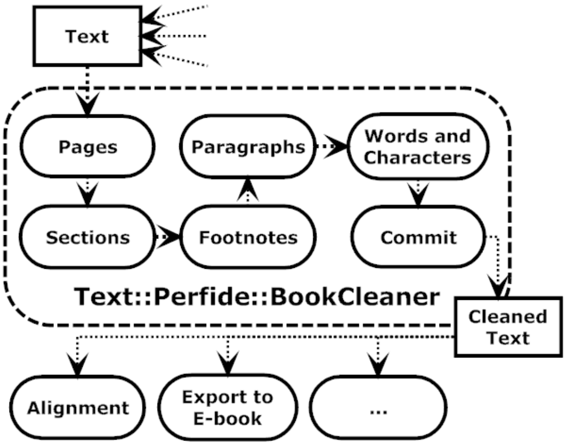


Figure 1: Pipeline of `T::P::BC`.

2.2.1 Pages

Page breaks, headers and footers are structural elements of pages that must be dealt with care when automatically processing texts. In the particular case of bitext alignment, the presence of these elements is often enough to confuse the aligner and decrease its performance. The headers and footers usually contain information about the book author, book name and section title. Despite the fact that the information provided by this elements might be relevant for some tasks, they should be identified, marked and possibly removed before further processing.

Below is presented an extract of a plain text version of *La Maison à Vapeur*, from Jules Verne, containing, from line 3 to 6, a page break (a footer, a page break character and a header, surrounded by blank lines).

```
1 rencontrer le nabab, et assez audacieux pour
2 s'emparer de sa personne.
3
4                    Page 3
5 ^LLa maison à vapeur          Jules Verne
6
7   Le faquir, - évidemment le seul entre tous
8 que ne surexcitât pas l'espoir de gagner la
```

The `0xC` character (also known as `Control+L`, `^L` or `\f`), indicates, in plain text, a page break. When present, these provide a reliable way to delimit pages. Page numbers, headers and footers should also be found near these breaks. There are books, however, which do not have their pages separated with this character.

Our algorithm starts by looking for page break characters. If it finds them, it immediately replaces them with a custom page break mark. If not, it tries to find occurrences of page numbers. These are typically lines with only three or less digits (four digits would occasionally get year references confused with page numbers), and a preceding and a following blank lines.

After finding the page breaks, the algorithm attempts to identify headers and footers: it considers the lines which are near each page break as candidates to headers or footers (depending on whether they are after or before the break, respectively). Then the number of occurrences of each candidate (normalized so that changes in numbers are ignored) is calculated, and those which occur more than a given threshold are considered headers or footers.

At the end, the headers and footers are moved to a standoff file, and only the page break mark is left in the original text.

A cleaned up version of the previous example is presented below:

```
est vrai qu'il fallait être assez chanceux pour
rencontrer le nabab, et assez audacieux pour
s'emparer de sa personne. _pb2_
  Le faquir, - évidemment le seul entre tous
que ne surexcitât pas l'espoir de gagner la
prime, - filait au milieu des groupes, s'arrêtant
```

2.2.2 Sections

There are several reason that justify the importance of delimiting sections in a book:

- automatically generate tables of contents;
- identify sections from one version of a book that are missing on another version (or example, it is common for some editions of a book to have an exclusive preamble or afterword which cannot be found in the other editions);
- matching and comparing the sections of two books before aligning them. This allows to assess the books *alignability* (compatibility for alignment), predict the results of the alignment and even to simply discard the worst sections;
- in areas like biomedical text mining, being able to detect the different sections of a document allows to search for specific information in specific sections (Agarwal and Yu, 2009);

Being an unstructured format, plain text by itself does not have the necessary means to formally represent the hierarchy of divisions of a book (chapters, sections, etc). As such, the notation used to write the titles of the several divisions is dictated by the personal choices of whoever transcribed the book to electronic format, or by their previous format and the tool used to perform the conversion to plain text. Additionally, the nomenclature used is often language-dependent (e.g. *Part One* and *Première Part* or *Chapter II* and *Capítulo 2*).

Below is presented the beginning of the first section of a Portuguese version of *Les Miserables*, from Vitor Hugo:

```
PRIMEIRA PARTE

FANTINE


^L LIVRO PRIMEIRO

UM JUSTO

O abade Myriel

Em 1815, era bispo de Digne, o reverendo Carlos
Francisco Bemvindo Myriel, o qual contava setenta
```

In order to help in the sectioning process, an ontology was built (see section 2.3.1), which includes several section types and their relation, names of common sections and ordinal and cardinal numbers.

The algorithm tries to find lines containing section names, pages or lines containing only numbers, or lines with Roman numerals. Then a custom section mark is added containing a normalized version of the original title (e.g. Roman numerals are converted to Arabic numbers).

The processed version of the example shown above follows:

```
_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 Myriel, o qual contava setenta
```

2.2.3 Paragraphs

When reading a book, we often assume that a new line, specially if indented, represents a new paragraph, and that sentences within the same paragraph are separated only by the punctuation mark and a space.

However, when dealing with plain text books, several other representations are possible: some books break sentences with a new line and paragraphs with a blank line (two consecutive new line characters); others use indentation to represent paragraphs; and there are even books where new line characters are used to wrap text. There are also books where direct speech (from a character of the story) is represented with a trailing dash; others use quotation marks to the same end.

The detection of paragraph and sentence boundaries is of the utmost importance in the alignment process, and a wrong delimitation is often responsible for a bad alignment.

In order to be able to detect how paragraphs and sentences are represented, our algorithm starts by measuring several parameters, such as the number of words, the number of lines, the number of empty lines and the number of indented lines.

Then several metrics are calculated based on these parameters, which are used to understand how paragraphs are represented, and the text is fixed accordingly.

Algorithm 1: Paragraph

```
Input: txt:text of the book
Output: txt:text with improved paragraph
elines ← ...calculate empty lines
lines ← ...calculate number of lines
words ← ...calculate number of words
plines ← ...number of lines with punctuation
$indent_i$ ← ...calculate indentation distrib
plr ← $\frac{plines}{lines}$                    // punctuated lines ratio

forall the $i \in dom(indent)$ do
    if $i > 10 \vee indent_i < 12$ then
        $remove(indent_i)$        // remove false indent

wpl ← $\frac{words}{lines}$                     // word per line
wpel ← $\frac{words}{1+elines}$                 // word per empty line
wpi ← $\frac{words}{indent}$                    // word per indent
if $wpel > 150$ then
    if $wpi \in [10..100]$ then
        Separate parag. by indentation
    if $wpl \in [10..100] \wedge plr > 0.6$ then
        Separate parag. by new lines
else
    Separate parag. by empty lines
```

2.2.4 Footnotes

Footnotes are formed by a call mark inserted in the middle of the text (often appended to the end of a word or phrase), and the footnote expansion (i.e. the text of the footnote). The placement of the footnote expansion shifts from book to book, but generally appears either at the end of the same page where its call mark can be found, or at a dedicated section at the end of the document.

Depending on the notation used to represent footnote call marks, these can introduce noise in the alignment (for example, if the aligner confuses them with sentence breaks). At best, the alignment might work well, but the resulting corpora will be polluted with the call marks, interfering with its further use. Footnote expansions are even more likely to disturb the process, as it is very unlikely that the matching footnotes are found at the same place in both books (the page divisions would have to be exactly the same).

Below is presented an example of a page containing footnote call marks and, at the end, footnote expansions, followed by the beginning of another page:

```
roi Charles V, fils de Jean II, 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.

^L Quel était en chemin l'étonnement de l'Ingénu!
je vous le laisse à penser. Il crut d'abord que
```

The removal of footnotes is performed in two steps: first the expansions are removed, then the call marks. The devised algorithm searches for lines starting with a plausible pattern (e.g. `<<1>>`, `[2]` or `^3`), and followed by blank lines. These are considered footnote expansions, are consequently replaced by a custom footnote mark and moved to a standoff file.

Once the footnote expansions have been removed, the remaining marks (following the same patterns) are likely to be call marks, and as such, removed and replaced by a custom normalized mark[1]

[1] The ideal would be to be able to establish the correspondence between the footnote call marks and their respective expansion. However, that feature is not specially relevant to the focus of this work, as the effort it would require is not compensated by its practical results.

After removing the footnote, the example given above would look like the following:

```
roi Charles V, fils de Jean II, auprès de la rue
Saint-Antoine, à la porte des Tournelles_fnr29_.
_fne8_
^L Quel était en chemin l'étonnement de l'Ingénu!
je vous le laisse à penser. Il crut d'abord que
```

2.2.5 Words and characters

At word and character level, there are several problems that can affect the alignment of two books: Unicode characters, translineations and transpaginations:

Unicode characters: often, Unicode-only versions of previously existing ASCII characters are used. For example, Unicode has several types of dashes (e.g. '-', '–' and '—'), where ASCII only has one (i.e. '-'). While these do not represent the exact same character, they are close enough, and their normalization allows to produce more similar results.

Glyphs: Some documents use glyphs (Unicode characters) to represent combinations of some specific characters, like *ﬁ* or *ﬀ*.

Translineation: translineation happens when a given word is split across two lines in order to keep the line length constant. If the two parts of translineated words are not rejoined before aligning, the aligner may see them as two separate words.

Transpagination: transpagination happens when a translineation occurs at the last line of a page, resulting in a word split across pages. However, the concept of page is removed by the first function, `pages`, which removes headers and footers and concatenates the pages. This means that transpagination occurrences get reduced to translineations.

The algorithm for dealing with translineations and transpaginations is available as an optional feature. This is mainly because different languages have different standard ways of dealing with translineations, and many documents use incorrect forms of translineation. As such, the user will have the option to turn this feature on or off.

The algorithm works by looking for words split across lines (word characters, followed by a dash, a newline and more word characters). When this pattern is found, the newline is removed and appended to the end of the word instead.

Dealing with Unicode-characters is performed with a simple search and replace. Characters which have a corresponding character in ASCII are directly replaced, while stranger characters are replaced with a normalized custom mark.

2.2.6 Commit

This last step removes the custom marks introduced by the previous functions and outputs a cleaned document ready to be processed. The only marks left are section marks, which can be helpful to guide the alignment.

There are situations where elements such as page numbers are important (for example, having page numbers in a book is convenient when making citations). However, removing these marks makes the previous steps irreversible. As such, this step is optional, and it is meant to be used only when a clean document is required for further processing.

2.3 Declarative objects

The first attempt at writing a program to clean books was a self-contained Perl script – all the patterns and logical steps were hardcoded into the program, with only a few command-line flags allowing to fine tune the process. When this solution was discarded for not being powerful enough to allow more complex processing, several higher level configuration objects were created: an ontology to describe section types and names, and domain-specific languages to describe configuration files.

These elements open the possibility to discuss the subject with people who have no programming experience, allowing them to directly contribute with their knowledge to the development of the software.

2.3.1 Sections Ontology

The information about sections relevant to the sectioning process is being stored in an ontology file (Uschold and Gruninger, 1996). Currently it contains several section types and their relation (e.g. a *Section* is part of a *Chapter*, an *Act* contains one or more *Scenes*), names of common sections (e.g. *Introduction*, *Index*, *Foreword*) and ordinal and cardinal numbers.

The recognition of these elements is language-dependent. As such, mechanisms to deal with several different languages had to be implemented. Some of these languages, even use a different alphabet (e.g. Russian), which lead to interesting challenges.

Several extracts of the sections ontology are presented below. The first example shows several translations and ways of writing *chapter*, and `NT sec` indicates *section* as a narrower term of *chapter*:

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

In the next example, *chapter* is indicated as a broader term of *sections*:

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

`BT _alone` allows to indicate that this term must appear alone in a line:

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

The number 3 and several variations. `BT _numeral` identifies this term as a number, and consequently might be used to numerate some of the other terms:

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

The typification of sections allows to define different rules and patterns to recognize each type. For example, a line beginning with *Chapter One* may be almost undoubtedly considered a section title, even if more text follows in the same line (which in many cases is the title of the chapter). On the other hand, a line beginning with a Roman numeral may also represent a chapter beginning, but only if nothing follows in the same line (or else century references, for example, would be frequently mistaken for chapter divisions).

Despite not yet being fully implemented, the relations between sections described in the ontology will allow to establish hierarchies, and, for example, search for *Scenes* after finding an *Act*, or even to classify a text as *play* in the presence of either.

Taking advantage of Perl being a dynamic programming language (i.e. is able to load new code at runtime), the ontology is being used to directly create the Perl code containing the complex data structures which are used in the process of sectioning. The ontology is in the ISO Thesaurus format, and the Perl module `Biblio::Thesaurus` (Simões and Almeida, 2002) is being used to manipulate the ontology.

2.3.2 Domain Specific Languages

The use of domain-specific languages (DSLs) will allow to have configuration files which control the flow of the process. Despite not yet implemented, it is planned to have `T::P::BC` reading a file describing external filters to be used at specific stages of the process (before or after any step).

This will allow to have calls to third-party tools, such as PDF to text converters or auxiliary cleaning scripts specific for a given type of files.

Both the sections ontology and DSLs help to achieve a better organization, allowing to abstract the domain layer from the the code and implementation details.

3 Evaluation

The evaluation of a tool such as `T::P::BC` might be performed by comparing the results of alignments of texts before and after being cleaned with `T::P::BC`.

In order to test `T::P::BC`, a set of 20 pairs of books from Shakespeare, both in Portuguese and English, was selected. From the 40 books, half (the Portuguese ones) contained both headers and footers.

The books were aligned using `cwb-align`, an aligner tool bundled with IMS Open Corpus Workbench (IMS Corpus Workbench, 1994-2002).

Pages, headers and footers

From a total of 1093 footers present in the documents, 1077 were found and removed (98.5%), and 1183 headers were detected and removed in a similar amount of total occurrences.

Translation units

The alignment of texts resulted in the creation of translation memories, files in the TMX format (Translation Memory eXchange (Oscar,)). Another way to evaluate `T::P::BC` is by comparing the number of each type of translation units in each file – either 1:1, 1:0 or 0:1, or 2:2 (the numbers represent how many segments were included in each variant of the translation unit). Table 1 summarizes the results obtained in the alignment of the 20 pairs of Shakespeare books.

Alignm. Type	Original	Cleaned	Δ%
0:1 or 1:0	8333	6570	-21.2
1:1	18802	23433	+24.6
2:1 or 1:2	15673	11365	-27.5
2:2	5413	4297	-20.6
Total Seg. PT	54864	53204	-3.0
Total Seg. EN	59744	51515	-13.8

Table 1: Number of translation units obtained for each type of alignment, with and without using `T::P::BC`[2].

The total number of 1:1 alignments has increased; in fact, in the original documents, 39.0% of the translation units were of the 1:1 type. On the other hand, the translation memory created after cleaning the files contained 51.3% of 1:1 translation units.

The total number of segments decreased in the cleaned version; some of the books used a specific theatrical notation which artificially increased the number of segments. This notation was normalized during the cleaning process with `theatre_norm`, a small Perl script created to normalize these undesired specific features, which lead to the smaller amount of segments.

The number of alignments considered bad by the aligner also decreased, from a total of 10 *"bad alignment"* cases in the original documents to 4 in the cleaned ones.

4 Conclusions and Future Work

Working with real situations, `Text::Perfide::BookCleaner` proved to be a useful tool to reduce text noise.

The possibility of using a set of configuration options helped to obtain the necessary adaptations to different uses.

Using `T::P::BC`, the translation memories produced in the alignment process have demonstrated an increase in the number of 1:1 translation units.

The use of ontologies for describing and storing the notation of text units (chapter, section, etc,) in a declarative human-readable notation was very important for maintenance and for asking collaboration from translator experts.

While processing the books from Shakespeare, we felt the need to have a *theater-specific* filter which would normalize, by example, the stage directions and the notation used in the characters name before each line.

This is a practical example of the need to allow external filters to be applied at specific stages of the cleaning process. This feature will be implemented soon in `T::P::BC`.

A special effort will be devoted to the recognition of tables of contents, which is still one of the main `T::P::BC` error causes.

Additionally, it is under consideration the possibility of using stand off annotation in the intermediary steps. That would mean to index the document contents before processing, and have all the information about the changes to be made to the document in a separate file. This would allow to have the original file untouched during the whole process, and still be able to *commit* the changes and produce a cleaned version.

`Text::Perfide::BookCleaner` will be made available at CPAN[3] in a very near future.

Acknowledgments

André Santos has a scholarship from Fundação para a Computação Científica Nacional and the work reported here has been partially funded by Fundação para a Ciência e Tecnologia through project Per-Fide PTDC/CLE-LLI/108948/2008.

References

Agarwal, S. and H. Yu. 2009. Automatically classifying sentences in full-text biomedical articles into Introduction, Methods, Results and Discussion. *Bioinformatics*, 25(23):3174.

Araújo, S., J.J. Almeida, I. Dias, and A. Simões. 2010. Apresentação do projecto Per-Fide: Paralelizando o Português com seis outras línguas. *Linguamática*, page 71.

[3] `http://www.cpan.org`

Evert, S. 2001. The CQP query language tutorial. *IMS Stuttgart*, 13.

Gupta, R. and S. Ahmed. 2007. Project Proposal Apache Tika.

IMS Corpus Workbench. 1994-2002. http://www.ims.uni-stuttgart.de/projekte/CorpusWorkbench/.

Lee, K.H., N. Guttenberg, and V. McCrary. 2002. Standardization aspects of eBook content formats. *Computer Standards & Interfaces*, 24(3):227–239.

Mattmann, Chris A. and Jukka L. Zitting. 2011. *Tika in Action*. Manning Publications Co., 1st edition.

Noonburg, D. 2001. xpdf: A C++ library for accessing PDF.

Okita, T. 2009. Data cleaning for word alignment. In *Proceedings of the ACL-IJCNLP 2009 Student Research Workshop*, pages 72–80. Association for Computational Linguistics.

Oscar, TMX. Lisa (2000) translation memory exchange.

Robinson, N. 2001. A Comparison of Utilities for converting from Postscript or Portable Document Format to Text. *Geneva: CERN*, 31.

Simões, A. and J.J. Almeida. 2002. Library::*: a toolkit for digital libraries.

UnmarkedSoftware. 2011. TextSoap: For people working with other people's text.

Uschold, M. and M. Gruninger. 1996. Ontologies: Principles, methods and applications. *The Knowledge Engineering Review*, 11(02):93–136.

Varga, D., P. Halácsy, A. Kornai, V. Nagy, L. Németh, and V. Trón. 2005. Parallel corpora for medium density languages. *Recent Advances in Natural Language Processing IV: Selected Papers from RANLP 2005*.

Véronis, J. 2000. From the Rosetta stone to the information society. pages 1–24.