An Algorithm for Sentence Recovery from PDF Files

UDC 81'322.2:004.912

ABSTRACT: The use of PDF documents in Natural Language Processing (NLP) became an almost daily activity for researchers in the field of computer linguistics and alike. Extracting plain text from PDF documents, with existing software tools, leads to severe distortion of sentence and paragraph structures, which is a huge problem for linguistically oriented research. In this paper, we present a novel algorithm for recovering sentences and paragraphs from PDF documents, called Sentence Recovery Algorithm or SR algorithm. The algorithm takes plain text extracted from a PDF document as an input, and tends to recover sentences from it. It takes into account cases like misinterpreted end of line, interruption of a sentence by tables or figures, problems occurred by hyphenation and so on. Beside describing and evaluating the algorithm, we present a use case for processing scientific articles originally given in PDF format, implemented in Java programming language.

KEYWORDS: Natural Language Processing, Language Resources, Java programming, PDF processing

Vesna Pajić
svesna@agrif.bg.ac.rs
University of Belgrade, Faculty of Agriculture

Staša Vujičić Stanković
stasa@matf.bg.ac.rs
University of Belgrade, Faculty of Mathematics

Miloš Pajić
paja@agrif.bg.ac.rs
University of Belgrade, Faculty of Agriculture

DATE OF SUBMISSION: 27 December 2014
DATE OF ACCEPTANCE: 18 April 2015

1 Introduction

Computer processing of texts written in natural languages (known as Natural Language Processing or NLP) is developing extensively in recent years. Its development is followed by its integration with other areas of computer science, such as text mining, information retrieval, information extraction, machine translation and others. All of these sub-areas of computing use text written in natural languages as an input, and process and transform it in different ways.

Textual resources differ in file formats they are stored in. Consequently, they must be processed in different ways. Ordinary text files (.TXT) and text files collected from the Internet in the form of hypertext (.HTML) will not be processed in the same manner. But despite the fact that they have different structures, formats such as .TXT, .HTML or .XML can be considered as text files, in the sense that the sequence of characters representing the text in them is continuous (undisturbed). It should be taken into account that HTML and XML files

have additional parts of text related to mark-up tags, and not natural language. However, clearly defined rules for insertion of HTML/XML tags allow their easy removal. Therefore, we consider that, from the point of NLP processing, these formats are equivalent. Nowadays, there is a number of software tools developed for efficient processing of these types of files. Some of them are UNITEX (Paumier, 2011), NooJ (Silberztein, 2003), GATE (Cunningham et al. 2002), different wrappers (Muslea et al. 1999; Kushmerick 2000.; Liu et al. 2000; Baumgartner et al. 2001.) and the like.

However, in recent years, especially on the web, PDF arises as a format for electronic document exchange. As an increasing number of researchers are using the web as a corpus, they are all faced with the processing of textual data in PDF format at some point. Unfortunately, processing of documents in PDF format using available tools suffers from numerous drawbacks. One of the main problems for linguistic processing of PDF text is violated original structure of sentence and text, i.e. sentence may be interrupted with the end of line character or some other objects. In this paper we present a new method for the automatic extraction of text files from PDF format into TXT format, which provides an opportunity to overcome the sentence's structure problem and allows further processing of texts by traditional methods and NLP tools.

In Section 2we describe the structure of PDF files, present some of the tools for converting PDF documents to text documents and describe the problems that arise in this process, from linguistic point of view. One of the biggest problems is violation of original sentence structure. In Section 3we present an algorithm for recovering the sentence structure to the level that allows the further processing of the text (SR algorithm). The implementation of this method in the Java programming language and an example of use are given in Section 4. In Section 5we evaluate the algorithm, showing it is very good from the standpoint of NLP. Finally, the conclusion has been given, together with some directives for future studies and activities in order to solve the presented problem.

2 Transforming PDF files into TXT files – state of the art

2.1 Portable Document Format (PDF)

Portable Document Format (PDF) is a file format invented by Adobe Systems[1] with the intention to represent documents independently of software and hardware platform. It is designed to preserve the original look of the document, i.e. the document will look the same on the screen and in print, regardless of what kind of computer or printer someone is using. Moreover, PDF files are highly compressed, allowing complex information to be downloaded from the web efficiently. As such, PDF became an open standard for electronic document exchange, maintained by the International Organization for Standardization (ISO)[2].

The structure and syntax of a PDF file is defined very strictly. A PDF document is a data structure composed of a small set of basic types of data objects, which are used to represent components of a PDF document: pages, fonts, annotations, and so forth. At the most fundamental level, a PDF file is a sequence of bytes. These bytes can be grouped according to the specific syntax rules. One or more groups are assembled to form higher level syntactic entities (objects), representing content of the document but also the way this content should be positioned and rendered on the pages of the document. For more details about PDF specification, we suggest reading the official reference.

Here, we are giving one example of PDF file content, which illustrates how PDF files store information about text. The text "ABC" is placed 10 inches from the bottom of the page and 4 inches from the left edge, using 12-point Helvetica. Corresponding part of the PDF document will look as follows:

```
BT
/F13 12 Tf
288 720 Td
(ABC) Tj
ET
```

1 http://www.adobe.com/devnet/pdf/pdf_reference.html
2 http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=51502

The five lines of this example perform these steps:

- Line 1 - Begin a text object.
- Line 2 - Set the font and font size, installing them as parameters in the text state. In this case, the font resource identified by the name F13 specifies the font externally known as Helvetica.
- Line 3 - Specify a starting position on the page, setting parameters in the text object.
- Line 4 - Paint the glyphs for a string of characters at that position.
- Line 5 - End the text object.

As seen in the example, there is much more information about a text string stored in the PDF document than it is needed for a linguistically oriented processing of the text. The majorpart of the information is concerning visual representation of the text, which is often irrelevant for NLP researches.

Therefore, in order to efficiently process a PDF document, one must get to know PDF syntax very well or rely on some existing software for converting PDF to TXT.

2.2 Software for converting PDF to TXT

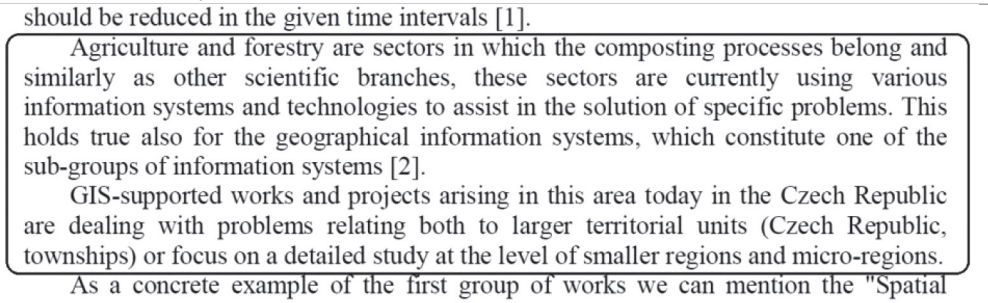
There are a lot of software tools for managing PDF files and most of them have an option to convert PDF files to plain text. With no intention to recommend any of them, we will here mention a few, just to illustrate the state-of-the-art in the field.

One of the most important is the Adobe's official software for managing PDF files. The current version is called Adobe Acrobat XI[3]. It allows editing and creating PDF documents, merging and combining files, protecting documents and converting PDFs to other formats, such as .TXT, .DOC, .HTML and others. ABBYY PDF Transformer 3.0[4] is anothercommercial software, with pretty much the same capabilities. It is a multilingual tool for easy converting of PDF file of any type into editable and searchable formats with the original layout and formatting retained. A number of free online tools are available on the web, such as SomePDF[5], ConvertPDF[6], ConvertPDFtoTXT[7] and others. There is also a Google's tool for viewing PDF files as plain HTML. Maybe the most successful tool that converts PDF is GATE's module for converting PDF to HTML. Although GATE does not have an option to convert to TXT directly, correctly recognized paragraphs in HTML format could be processed easily as text.

For developers, there are APIs for almost every programming language, which helps in managing PDF files from programming code. The PDFClown[8] and IcePDF[9] are worth mentioning, since they are highly functional, bug-free, good documented and easy to use. PDF Clown is a free open source API, written as a class library in multiple languages (Java™ 6 and C#/.NET 4.0). IcePDF is an open source Java PDF engine for viewing, printing, and manipulating PDF documents. It can be used as a standalone open source Java PDF viewer, or can be easily embedded into any Java application. Beyond PDF document rendering, it can be used for PDF to image conversion, PDF search or PDF text and/or image extraction.

No matter which one of software tools is used for conversion, same problems occur from a linguistics perspective. Each tool will disrupt the structure of sentences and paragraphs during the conversion of PDF files. The reason for this is not the imperfection of existing tools, but the nature of the PDF document, which primarily stores visual components necessary to provide the same layout of the document, regardless of computer systems and components used by the user.

3 http://www.adobe.com/rs/products/acrobat.html
4 http://pdftransformer.abbyy.com/
5 http://download.cnet.com/Some-PDF-to-Txt-Converter/3000-2079_4-10836740.html
6 http://convertonlinefree.com/PDFToTXTEN.aspx
7 http://www.convertpdftotext.net/
8 http://www.stefanochizzolini.it/en/projects/clown/index.html
9 http://www.icesoft.org/java/projects/ICEpdf/overview.jsf


should be reduced in the given time intervals [1].

Agriculture and forestry are sectors in which the composting processes belong and similarly as other scientific branches, these sectors are currently using various information systems and technologies to assist in the solution of specific problems. This holds true also for the geographical information systems, which constitute one of the sub-groups of information systems [2].

GIS-supported works and projects arising in this area today in the Czech Republic are dealing with problems relating both to larger territorial units (Czech Republic, townships) or focus on a detailed study at the level of smaller regions and micro-regions.

As a concrete example of the first group of works we can mention the "Spatial

Figure 1. An excerpt from the PDF document (the text is a part of Journal of Agricultural Engineering, Volume 4, 2012).

2.3 The conversion problems

Let us look at the process of converting document from PDF to TXT format more closely. As we mentioned in Section 2.2, this process is similar regardless the software tool used. The input of a conversion process is a document in PDF format (denoted as File.PDF), and the output is a file in TXT format created by a software for conversion (denoted as File.TXT). In the following examples we use IcePDF for conversion, if not stated otherwise. There are some issues of the conversion process, which can be viewed as problems for linguistically oriented processing.

2.3.1 Inserting the end-of-line (EOL)

The end-of-line (EOL) is one or more characters, used to indicate the end of a line or a paragraph in textual files. Which character will be used for EOL depends on the software platform. On Windows OS, carriage return (CR) followed by line feed (LF) is used (also denoted as '\r\n' or 0x0D0A). For visual representation, when needed, the *pilcrow* character (¶) is used.

In most cases of PDF to TXT conversion, the EOL characters are inserted at positions where a line of text visually breaks, not only where a paragraph actually ends. In that way, information about paragraphs and sentences is lost; a paragraph from File.PDF (Figure 1) is converted into several paragraphs (Figure 2) in File.TXT (a paragraph per each visual line of the text).

2.3.2 Interrupting a paragraph with an object

PDF files often have a lot of different objects inserted in the text in different ways. Those can be tables, figures, graphs, formulas, page objects (header, footer, page numbers etc.) and so on. If there is an object (as a table, figure, header and so on) inserted in a paragraph in File.PDF (Figure 3), the text sequence in File.TXT is broken with some additional lines (Figure 4), which are the products of converting the object.

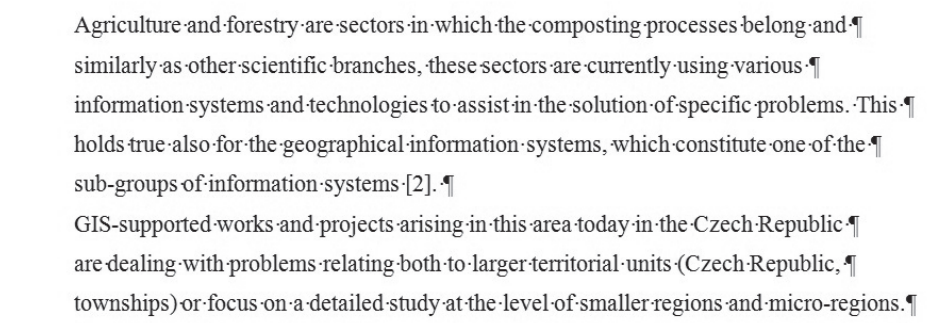
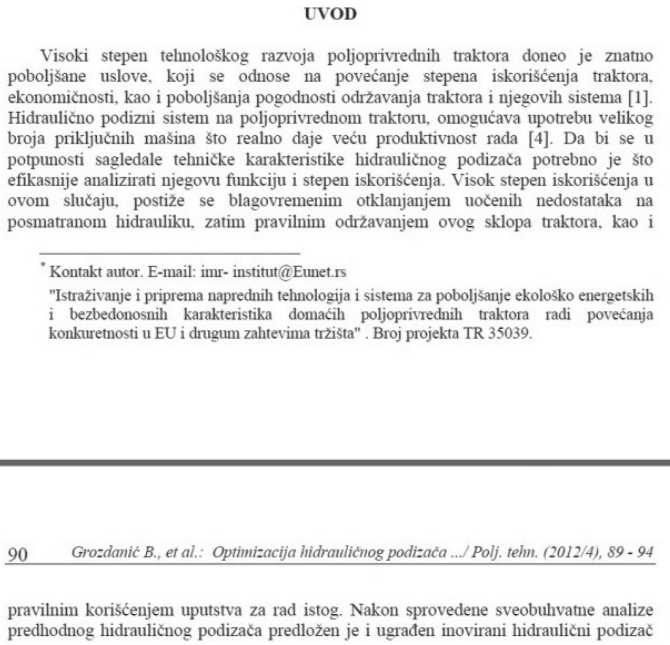

Agriculture·and·forestry·are·sectors·in·which·the·composting·processes·belong·and·¶
similarly·as·other·scientific·branches,·these·sectors·are·currently·using·various·¶
information·systems·and·technologies·to·assist·in·the·solution·of·specific·problems.·This·¶
holds·true·also·for·the·geographical·information·systems,·which·constitute·one·of·the·¶
sub-groups·of·information·systems·[2].·¶
GIS-supported·works·and·projects·arising·in·this·area·today·in·the·Czech·Republic·¶
are·dealing·with·problems·relating·both·to·larger·territorial·units·(Czech·Republic,·¶
townships)·or·focus·on·a·detailed·study·at·the·level·of·smaller·regions·and·micro-regions.·¶

Figure 2. The text marked in Figure 1after converting to the TXT format; the hidden characters, such as space and end of paragraph are shown.



UVOD

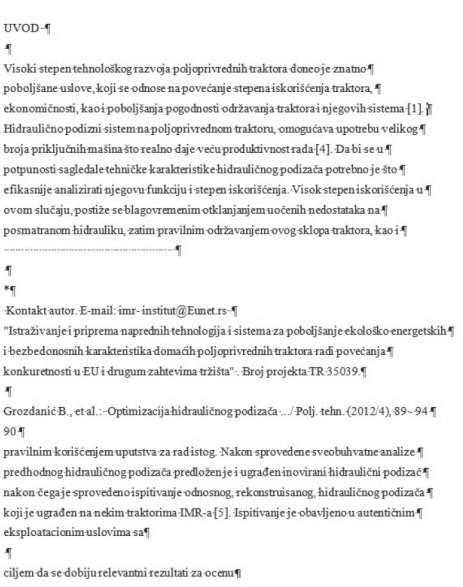
Visoki stepen tehnološkog razvoja poljoprivrednih traktora doneo je znatno poboljšane uslove, koji se odnose na povećanje stepena iskorišćenja traktora, ekonomičnosti, kao i poboljšanja pogodnosti održavanja traktora i njegovih sistema [1]. Hidraulično podizni sistem na poljoprivrednom traktoru, omogućava upotrebu velikog broja priključnih mašina što realno daje veću produktivnost rada [4]. Da bi se u potpunosti sagledale tehničke karakteristike hidrauličnog podizača potrebno je što efikasnije analizirati njegovu funkciju i stepen iskorišćenja. Visok stepen iskorišćenja u ovom slučaju, postiže se blagovremenim otklanjanjem uočenih nedostataka na posmatranom hidrauliku, zatim pravilnim održavanjem ovog sklopa traktora, kao i

* Kontakt autor. E-mail: imr- institut@Eunet.rs

"Istraživanje i priprema naprednih tehnologija i sistema za poboljšanje ekološko energetskih i bezbedonosnih karakteristika domaćih poljoprivrednih traktora radi povećanja konkuretnosti u EU i drugum zahtevima tržišta" . Broj projekta TR 35039.

pravilnim korišćenjem uputstva za rad istog. Nakon sprovedene sveobuhvatne analize predhodnog hidrauličnog podizača predložen je i ugrađen inovirani hidraulični podizač

Figure 3. The end of one and the beginning of the next page; the paragraph is split by the footnote and the page header.



UVOD·¶
¶
Visoki·stepen·tehnološkog·razvoja·poljoprivrednih·traktora·doneo·je·znatno·¶
poboljšane·uslove,·koji·se·odnose·na·povećanje·stepena·iskorišćenja·traktora,·¶
ekonomičnosti,·kao·i·poboljšanja·pogodnosti·održavanja·traktora·i·njegovih·sistema·[1].·¶
Hidraulično·podizni·sistem·na·poljoprivrednom·traktoru,·omogućava·upotrebu·velikog·¶
broja·priključnih·mašina·što·realno·daje·veću·produktivnost·rada·[4].·Da·bi·se·u·¶
potpunosti·sagledale·tehničke·karakteristike·hidrauličnog·podizača·potrebno·je·što·¶
efikasnije·analizirati·njegovu·funkciju·i·stepen·iskorišćenja.·Visok·stepen·iskorišćenja·u·¶
ovom·slučaju,·postiže·se·blagovremenim·otklanjanjem·uočenih·nedostataka·na·¶
posmatranom·hidrauliku,·zatim·pravilnim·održavanjem·ovog·sklopa·traktora,·kao·i·¶
----------¶
¶
*¶
·Kontakt·autor.·E-mail:·imr-·institut@Eunet.rs·¶
"Istraživanje·i·priprema·naprednih·tehnologija·i·sistema·za·poboljšanje·ekološko·energetskih·¶
i·bezbedonosnih·karakteristika·domaćih·poljoprivrednih·traktora·radi·povećanja·¶
konkuretnosti·u·EU·i·drugum·zahtevima·tržišta"·.·Broj·projekta·TR·35039.·¶
¶
Grozdanić·B.,·et·al.:·-Optimizacija·hidrauličnog·podizača·.../·Polj.·tehn.·(2012/4),·89·-·94·¶
90·¶
pravilnim·korišćenjem·uputstva·za·rad·istog.·Nakon·sprovedene·sveobuhvatne·analize·¶
predhodnog·hidrauličnog·podizača·predložen·je·i·ugrađen·inovirani·hidraulični·podizač·¶
nakon·čegaje·sprovedeno·ispitivanje·odnosnog,·rekonstruisanog,·hidrauličnog·podizača·¶
koji·je·ugrađen·na·nekim·traktorima·IMR-a·[5].·Ispitivanje·je·obavljeno·u·autentičnim·¶
eksploatacionim·uslovima·sa¶
¶
ciljem·da·se·dobiju·relevantni·rezultati·za·ocenu¶

Figure 4. The converted text.

2.3.3 The hyphenation problem

Hyphenation is a process of inserting hyphen character ("-") into words, usually between two syllables, in order to break a line of text. If words in File.PDF are hyphenated, then those words will not convert into adequate word forms in File.TXT; for example, if the word "*metabolic*" is hyphenated as "*meta-bolic*", it will be converted into sequence "*meta-*", EOL character and "*bolic*". The simplest approach would be to remove EOL character and character "-", and to produce one word ("*metabolic*"). But there are cases like "*amino-glycosides*" where hyphen character should remain.

2.3.4 The wrong character interpretation

Since PDF format is focused on displaying documents and preserving the same look over different platforms, the common problem during conversion is inadequate character representation; this problem is more obvious when processing texts with non-Latin characters. Here we need to mention that the following problems occurred while using ABBYY PDF Transformer, while IcePDF converted characters correctly. Nevertheless, it is important to emphasize this problem, since a lot of researchers will face it at some point.

For example, the most frequent problems we noted are changing Cyrillic letter **з** with the glyph

з, Cyrillic letter **и** with Latin letter **u** (the italic form of letter **и** is ***u***), Cyrillic letter **у** with Latin letter **y**. Furthermore, there is a problem with converting two characters, such as **fl** into one glyph **ﬂ** (as in "*single flagellum*") and many others. Commonly, it is very hard to notice these kinds of misinterpretations of characters in the first place, since incorrectly converted characters look the same.

3 The Algorithm for Sentence Recovery (SR Algorithm)

In majority of NLP techniques and methods, a sentence is a basic unit of text processing. Therefore, the format of file obtained after conversion from PDF to TXT (we called it File.TXT) is inadequate for further linguistic processing and analyzing. Since PDF documents increasingly dominate as a way of storing documents, especially on the WWW, the additional processing and preparation of those files is necessary in order to process them with NLP tools. This job can be time consuming if done by humans, because collections of PDF documents are usually huge.

Having the same problems in our research and trying to overcome them, we developed an algorithm that automates the process of preparing the texts for further processing. It decreases the need for human engagement in the preparation process. We have adapted the algorithm so it can be used in many similar situations by other researchers as well. It is simple enough to be widely used by others, and precise enough to be used in NLP processing with high reliability. Moreover, the algorithm itself is language independent, since it is based on statistical properties of text.

The algorithm is primarily developed for processing documents that consist mostly of text, having the form similar as printed documents such as scientific papers, books or newspapers. Although they can contain different graphical objects (photos, graphs, tables...), the information in the form of natural language text dominates in these documents. Moreover, the text is organized into paragraphs, with sporadically inserted headings and titles. The rest of PDF files, such as PPT slides or some catalogues with a lot of images, cannot be processed with SR algorithm efficiently.

3.1 The basic flow of the SR Algorithm

The input of the SR algorithm is a text obtained after initial conversion of a textual document from PDF to TXT format, performed by any existing software. The properties of such texts are already described in previous sections, but the main property is that the structure of sentences and paragraphs is disrupted in some way, so the processing of the text is not possible with NLP tools.

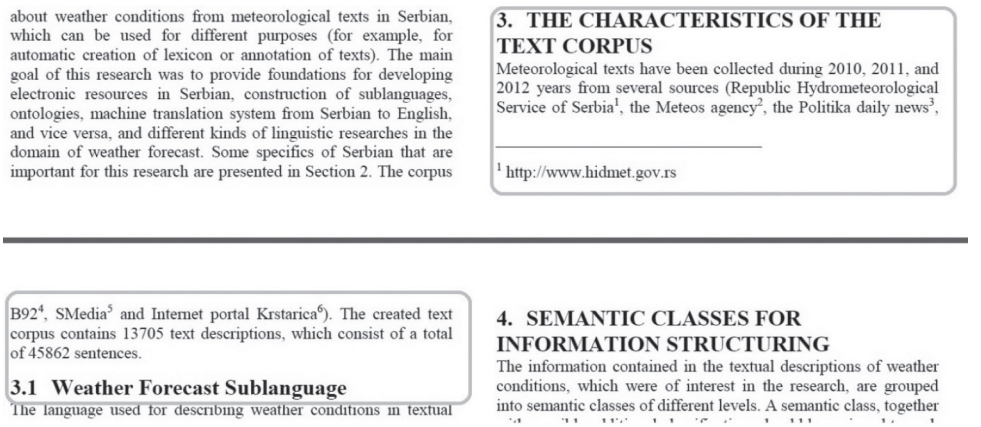
Since rhetorical structure of the text is violated, the input text is seen as a sequence of text lines. SR algorithm tries to identify each line as one of the following:

- a heading line or part of a heading;
- a beginning, a central part or an ending of a text paragraph;
- a caption of an object (table caption, figure caption and so on);
- a part of a converted object (for example, parts of a table or converted formulas);
- a page element, such as a page header, page numbers and so on.

After identification of a line, SR algorithm takes some actions based on the identified form. Those actions can vary depending on the final purpose of text processing. For example, if someone is interested only in studying the language, then maybe tables are irrelevant to them and they can be omitted from the output file. On the other hand, if someone is doing information extraction from the text, the tables may have crucial significance, and they will remain in the output file, so they can be further processed.

The basic form of SR algorithm, presented in this paper, will do the following:

- a heading line or subsequent heading lines will be converted into one paragraph;
- EOL characters will be removed from the lines recognized as a beginning or a central part of a paragraph, with special processing of hyphenated words;
- captions of objects will remain as separate paragraphs;
- parts of converted objects will be removed from the output file;


about weather conditions from meteorological texts in Serbian, which can be used for different purposes (for example, for automatic creation of lexicon or annotation of texts). The main goal of this research was to provide foundations for developing electronic resources in Serbian, construction of sublanguages, ontologies, machine translation system from Serbian to English, and vice versa, and different kinds of linguistic researches in the domain of weather forecast. Some specifics of Serbian that are important for this research are presented in Section 2. The corpus

3. THE CHARACTERISTICS OF THE TEXT CORPUS

Meteorological texts have been collected during 2010, 2011, and 2012 years from several sources (Republic Hydrometeorological Service of Serbia[1], the Meteos agency[2], the Politika daily news[3],

[1] http://www.hidmet.gov.rs

B92[4], SMedia[5] and Internet portal Krstarica[6]). The created text corpus contains 13705 text descriptions, which consist of a total of 45862 sentences.

3.1 Weather Forecast Sublanguage

The language used for describing weather conditions in textual

4. SEMANTIC CLASSES FOR INFORMATION STRUCTURING

The information contained in the textual descriptions of weather conditions, which were of interest in the research, are grouped into semantic classes of different levels. A semantic class, together

Figure 5. A part of a PDF document (REF); the processing of the text with the gray frame around it will be explained.

- page elements will be removed from the output file.

Here follows an example of an input text and its form after the processing. Figure 5 shows an excerpt from a PDF file. The further processing example focuses on the part of the text with gray frame around it. It consists of one heading, a paragraph spreading across two pages with a footnote inserted, and one subheading.

After converting the PDF file with conventional software (in this particular case we used IcePDF, because of its properties discussed in 2.3.4), the 11 text lines shown on Figure 6 are obtained.

3.·THE·CHARACTERISTICS· OF·THE·¶
TEXT·CORPUS·¶
Meteorological texts have been collected during 2010, 2011, and ¶
2012 years from several sources (Republic Hydrometeorological ¶
Service of Serbia1, the Meteos agency2, the Politika daily news3, ¶
.....¶
1 http://www.hidmet.gov.rs¶
·B924, ·SMedia5 ·and ·Internet ·portal ·Krstarica6). ·The ·created ·text ·¶
corpus contains 13705 text descriptions, which consist of a total ¶
of 45862 sentences. ¶
3.1 Weather Forecast Sublanguage ¶

Figure 6. The text after initial converting from PDF to TXT; hidden characters (EOL and spaces) are shown to demonstrate the real content of the text.

Ideally, the SR algorithm should process these lines in the following manner:

- Lines 1 and 2 should be recognized as a heading, and merged into one paragraph (we will do this by analyzing the last characters of lines, the case of the letters, and the length of the lines);
- Lines 3, 4, and 5 should be recognized as one paragraph and merged together; the EOL characters will be removed from the end of lines and spaces will be inserted instead;
- Line 6, containing only spaces and EOL should be removed totally (we will do this by determining that the previous paragraph have not finished yet, so the Line 6 must be a part of some inserted object – a footnote in this case);
- Line 7 should be recognized as a footnote and removed entirely (we will do this by analyzing its length);
- Line 8, 9 and 10 should be recognized as the rest of the previous paragraph and merged together with spaces, removing EOL characters, except in the Line 10, where the paragraph actually ends;
- Line 11 should be recognized as a heading and remain as it is, i.e. one paragraph.

After processing, the text should look as in Figure 7.

3.·THE·CHARACTERISTICS·OF·THE·TEXT·CORPUS·¶

Meteorological texts have been collected during 2010, 2011, and 2012 years from several sources (Republic Hydrometeorological Service of Serbia1, the Meteos agency2, the Politika daily news3, B924, SMedia5 and Internet portal Krstarica6). The created text corpus contains 13705 text descriptions, which consist of a total of 45862 sentences. ¶

3.1·Weather·Forecast·Sublanguage·¶

Figure 7. The text after processing with SR algorithm; hidden characters (EOL and spaces) are shown to demonstrate the real content of the text.

3.2. Some assumptions

During our research we processed a number of PDF files, mostly scientific articles and encyclopedia texts. Since they are primarily made for paper printing, they all have the similar form and some similar features. Those features helped a lot in designing the algorithm. Therefore, we retained them as assumptions a document must satisfy in order to use SR algorithm for its successful processing. The processing with SR algorithm is possible for documents that do not satisfy these conditions, but with decreased overall efficiency and accuracy.

The assumptions are:

1. The document consists mostly of text, organized into paragraphs and headings, with objects only sporadically inserted.
2. Text paragraphs that are not headings have the same font size when displayed, so text lines belonging to paragraph have approximately equal numbers of characters in length in most of the document. This length corresponds to the length of one column of text in a document.
3. Headings can be distinguished from the rest of the text, not only by formatting in the original document, but also by other properties such as ALL CAPS, Title Case or having a new line before and after.
4. Headings are mainly shorter than paragraph lines, i.e. they have fewer characters than a line in a text paragraph.

Based on assumption 2, we analyzed the length of text lines in a document. Although documents can vary in formats, texts such as scientific articles, newspapers articles or books will have similar distribution of lengths of text lines. As an example, the distribution of three different documents is presented in Figure 8a, 8b, and 8c.

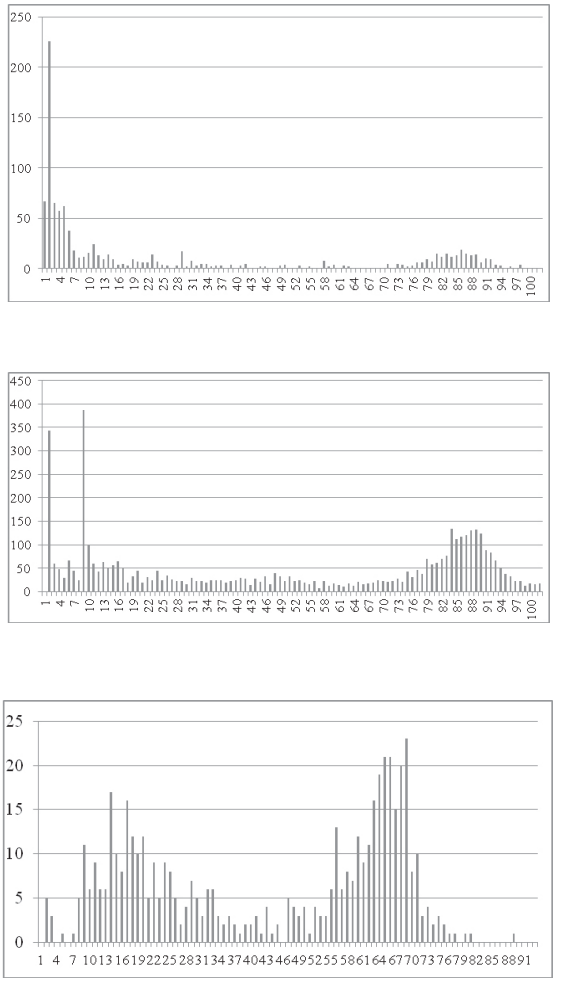


Figure 8 Distribution of the number of lines having different length; vertical axis represents number of text lines in a document and horizontal axis represents number of characters per line; 8.a and 8.b represent distributions of two different one column documents; 8.c represents the distribution of a two columns document.

The documents whose distribution is shown in Figures 8.a and 8.b are one column text documents and document whose distribution is shown in Figures 8.c is a two column text document. During processing a number of documents (examples given here show this as well), we observed that the most of the lines will be distributed around

two values. The first of them will be in the left side of the graph, near zero length, and it can be treated as a short line mean. It is more obvious in the case of first two documents and it inflects existence of many text lines with length 0 or less than 10 (these are usually empty lines, parts of converted objects such as tables, or lines containing just a page number). The second value will be positioned in the right side of the distribution graph and this value represents the mean of long lines.

This second value is of great importance for SR algorithm, since it describes the length of lines belonging to a text paragraph, i.e. the width of the text column. We will call it here *columnwidth* (CW) *value*. CW value will vary among different documents depending on a page size of a document, its margins or font size. For example, just by looking in histograms given in Figure 8, we can observe that the document represented in Figure 8.a has CW value around 85 characters, and the document from Figure 8.b has CW around 88 characters. CW value for the third example document is around 65, indicating that this document has narrower columns than the first two (this was due to splitting text into two columns on a page).

3.3 Calculating CW value

SR algorithm uses CW value for deciding if a text line should be a part of a text paragraph or a heading. Since it differs for different documents, it is necessary to calculate CW value for each document being processed.

A text document being processed with SR algorithm, as we already showed, can be seen as a sequence or an array of text lines. In that manner, we can represent a document D as $D=\{t_i, i=1..|D|\}$, where |D| is a number of text lines in the document D and t_i is a text line. Each line t_i has its length, i.e. the numbers of characters in the line, without final EOL. We will denote this length as $l(t_i)$. Let m be $max_D l(t_i)$, that is the maximum length of lines in a document.

A distribution of line lengths (DL) will be an array of integers $DL = \{x_j N | j=0\ldots m\}$, such that there is x_j lines in the document having the length j (Figure 8). As showed in Section 3.2, DL has a bimodal distribution and we need to find its right mode.

We first calculate the average length l_{avg} of all lines in a document as

$$l_{avg} = \frac{\sum_{i \in 1..|D|} l(t_i)}{|D|} \quad l_{avg} = \frac{\sum_{i \in 1..|D|} l(t_i)}{|D|}$$

$$l_{avg} = \frac{\sum_{i \in 1..|D|} l(t_i)}{|D|} \quad l_{avg} = \frac{\sum_{i \in 1..|D|} l(t_i)}{|D|} .$$

This average line length is important because if we split DL histogram with l_{avg}, the CW value will remain in the right part of histogram. Then, we will calculate CW as a length that maximum number of lines in the right part of the histogram have:

$CW = \{i \mid x_i = max\{x_j \mid j \geq l_{avg}\}\}$.

If we calculate CW with the above formula, documents represented in Figure 8 will have CW values 85, 83 and 68 respectively.

3.4 The design and schema of SR Algorithm

The main algorithm's task is to recover sentences that are violated during conversion. Therefore, keeping or merging together parts of the text that contain whole sentences is essential, i.e. a sentence should not be split in two by EOL character. It would be ideally to transform a part of the text that represents a paragraph in original PDF document into one paragraph in final TXT file, but it is possible that SR algorithm transforms it into two or more paragraphs. We will not consider this as an error, as long as each sentence begins and ends in the same paragraph, i.e. there are no sentence that spreads across two paragraphs.

In the light of this, the algorithm distinguishes three main types of input text lines: empty lines (EL), containing only a zero or more spaces and EOL character; finished lines (FL), ending with one of the character from the set F={. , ?, !} followed by EOL: and unfinished lines (UL), not ending with one of the character from the set F={. , ?, !} and EOL.

Processing of EL and FL is trivial; EL will be deleted from the output file and the algorithm will add EOL character at the end of FL. The UL lines, on the other hand, need to be processed and analyzed for several possibilities (whether they are a part of a heading, a paragraph or a converted object).

The basic flow of SR algorithm is shown in Figure 9. While reading lines from the file, the

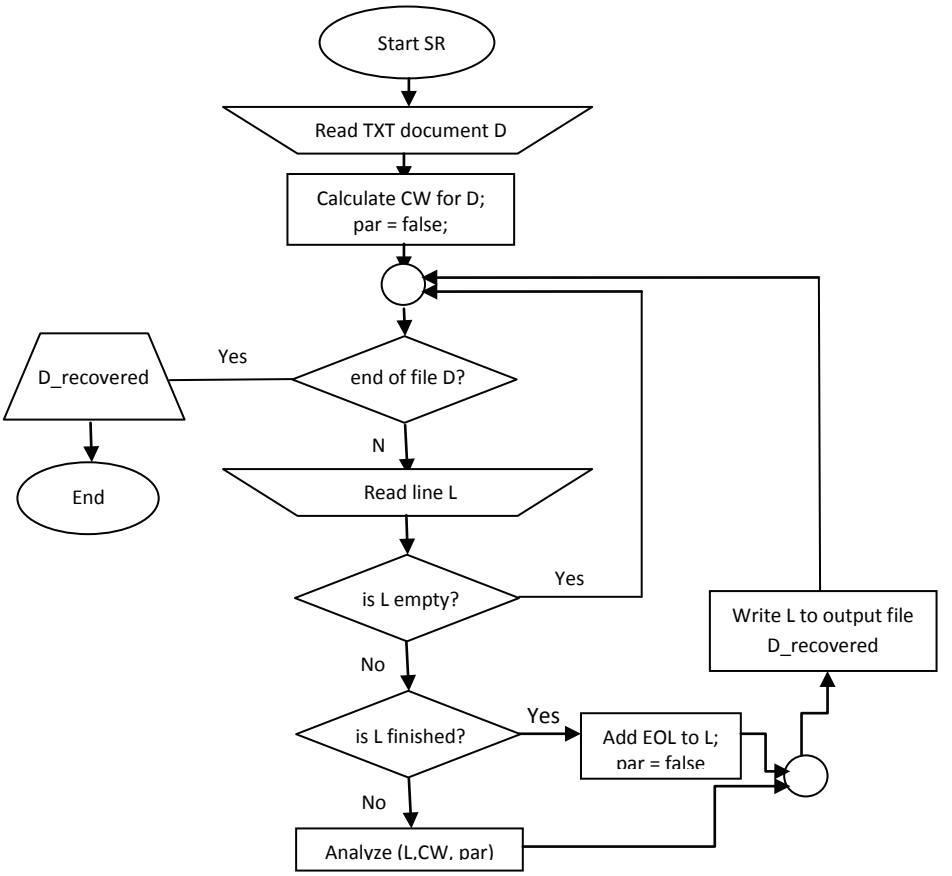


Figure 9. The basic flow schema of the SR algorithm.

algorithm keeps track of a paragraph, i.e. it sets the boolean value *par* to TRUE when a paragraph starts and to FALSE, when the paragraph ends. This value is important for the analysis of UL lines.

The analysis of UL lines (denoted as Analyze(L,CW,par)) is shown in Figure 10. It is based on comparing the length of an UL line with the CW value. Depending on the structure of a document (whether it is a one column or a two column text), line lengths belonging to text paragraphs differ in more or less number of characters. The initial algorithm is designed to tolerate up to 10% of CW value; for example, if CW value is 85 characters, lines having the length between 77 and 93 characters will be a part of a text paragraph. This value can be changed in order to better reflect the structure of a document, if needed. Then, the comparison result indicates whether an UL line is a part of a text paragraph or not.

It is important to notice here that the analysis of UL lines depends also on the value *par*, i.e. whether a paragraph has already started or not. In that way, the algorithm can distinguish a heading from a converted object line, since heading cannot be inserted into a paragraph. If titles and headings have additional properties (for example, they are written in Title Case or ALL CAPS), the additional processing can be done in order to determine if some line belongs to a heading.

The processing of hyphenated words can be done in a primitive way, just by deleting the ending hyphen ("-"), and merging with next line. If some additional resources are available to the user, such as lexicons, electronic dictionaries or so, it is possible to deal with hyphenated words in a more sophisticated manner, where the algorithm could distinguish a hyphenated word from a compound word. Here, we used the first way of dealing with hyphens.

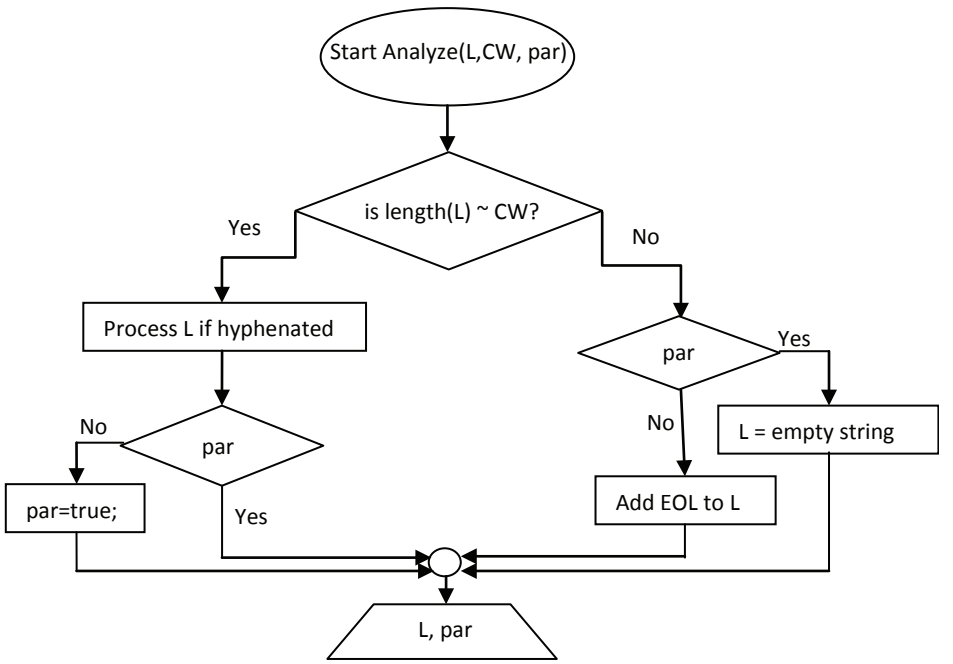


Figure 10. The analysis of UL lines

4 Use case – A Java implementation example

During our research, we used SR algorithm several times, processing different types of documents, from scientific articles to novels. For that purposes, we create a Java class, called *SentenceRecovery*, which implements SR algorithm. Since we had a lot of PDF documents already converted to TXT with some other tools, the *SentenceRecovery* class for now serves only as a standalone class, but it can be re-designed and integrated as an extension of IcePDF classes for processing PDF file.

The members of *SentenceRecovery* class are shown on Figure 11. The object attribute linesis an array of strings, containing lines from a document that need to be processed (EL, FL and UL lines). The indicator par is used to keep track of a paragraph, whether it has started and is (or is not) finished yet. The object attribute CW keeps the CW value of a document. The variable eps represents the allowed deviation from CW value. The default value for eps is 10%. The object attribute outTekststores the document text after processing.

Initially, after creating an object of *SentenceRecovery* class, the first step is to populate array lines. It is usually done by reading a file containing a document and method readLines() is used for that purpose. Then, it is necessary to calculate the CW value for the document using method calculateCW(). The both methods are called within the constructor of the class.

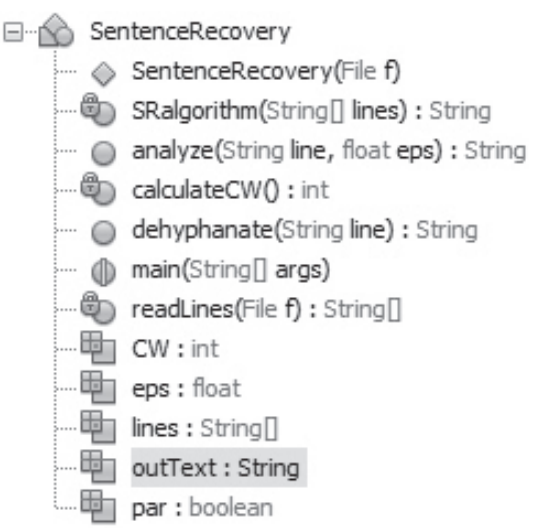


Figure 11. Members of SentenceRecovery class.

Methods SRalgorithm() and analyze() are direct implementations of algorithms presented already in Section 3.4 (Figures 9 and 10).

Here is an example of Java code that uses SentenceRecovery class:

```
File dir = new File(someDir);
File[] files = dir.listFiles();
for (File f:files){
    SentenceRecoverysr = new SentenceRecovery(f);
    try{
        FileOutputStream out =
            newFileOutputStream(f.
            getAbsolutePath() + "._SRalg" );
        out.write(sr.outText.getBytes());
        out.close();
    } catch (Exception e) {
        e.printStackTrace();
        System.out.println(sr.outTekst);
    }
}
```

The above code takes a list of files from a directory and generates recovered text for each of them. The text is then saved in a new file within the same directory. In that way, with just a few lines of code, a number of files can be processed.

5 Experimental evaluation and the analysis of results

In order to test and evaluate the SR algorithm, we conducted several experiments. The experiments were designed taking into account some previously spotted or assumed setbacks. The most important thing was to evaluate efficiency of correctly added EOL characters.

The analysis of lines and decision whether to put an EOL to the end of a line or not depends on the structure of documents. Therefore, we processed three different types of documents, each in the same way, and compared the results. The test documents were a scientific journal with one column text, a scientific journal with two columns text and a novel.

The first experiment was on comparing our results with the results obtained from another software tool that tries to determine the beginning and the ending of paragraphs in PDF text. The second experimentwas ondetermining the optimal eps value, used for comparing a length of a line with CW value.

5.1. Comparison with existing software

There is a lot of software tools for converting PDF file to TXT format (or similar, such as HTML or XML). However, not so many of them try to recover rhetorical structure of documents. Instead, focusing just on visual aspects, they all embed EOL characters at the visual end of a line, breaking the structure of sentences in that way.

We found that, among common tools having the capability to process PDF files, only GATE has some sort of sentence/paragraph recovery. Although the GATE as well inserts EOL characters at positions where lines visually break, it also tries to recognize the beginning and the ending of paragraphs and inserts tags and at that positions. Therefore, we chose GATE as software for compare with.

In order to decrease the influence of document structure on the converting process, we processed documents with three different structures. The first document is a scientific journal with one column text, the second is a scientific journal with two columns text and the third is a novel. The first two documents have more headings comparing with the third. Moreover, in the first two documents text paragraphs are often split in two by different objects such as tables, figure or math formulas, while there are no objects in the third document. Instead, the third document has a lot of short paragraphs that are parts of dialogues.

As already mentioned in Section 3.4, the basic task was to preserve the structure of sentences, that is, not to have a sentence that spreads across two paragraphs. The main purpose of our research was to prepare texts for further analysis with linguistically oriented software. Since these kinds of software tool use a sentence as a unit for processing, we analyzed results based on the number of broken sentences that remained after processing. The results of the comparison are presented in Table 1. Both, absolute and relative counts are given.

Document structure	Tool for recovering text	Sentences that remained broken after processing	
		Total count	%
One column scientific journal	GATE	10	10%
	SR algorithm	3	3%
Two column scientific journal	GATE	12	15%
	SR algorithm	6	8%
Novel	GATE	323	23%
	SR algorithm	0	0%

Table 1. The results of performance comparison between SR algorithm and the GATE

5.2 Experimenting with *eps* value

The *eps* value is used in SR algorithm as an allowed deviation from CW value, i.e. it represents the number of characters a line from a text paragraph can differ from the average paragraph length. Its default value in SR algorithm is 0.1, which is determined empirically.We tried to experiment with this value and to see if there is a significant difference in efficiency.

Based on the design of the SR algorithm, it is clear that increasing the *eps* value will result in higher tolerance of a line length, so there will be less cases in which a UL line will be incorrectly recognized as either an embedded object (and thereforedeleted) or an end of the paragraph (and therefore split across two paragraphs). On the other hand, some long headings will be recognized as parts of text paragraphs, so it is possible to have some text sequences with improper syntax and semantics. Decreasing the *eps* value will have opposite effects.

We used values 0.3, 0.1 and 0.05 for testing. The testing was conducted on several different documents, with different rhetorical structures. Here are some conclusions and recommendations based on them.

For processing texts such as novels, where the main text is written in one column and is not interrupted with objects such as tables, footnotes, images and alike, the *eps* value should be increased to 0.3, especially if the headings are not frequent in the text and are relatively short comparing to the rest of the text.

For processing texts such as journal articles or encyclopedias, the *eps* value should be smaller, depending on the structure of the text. If the text is written in two or three columns, the differences in a number of characters per line are smaller between the lines, so the *eps* value should be decreased to 0.05.

6 Conclusion and future work

The presented SR algorithm and its implementation have a practical application in linguistically oriented research and text processing. Researchers can have a clear benefit from the algorithm, since it decreases significantly the amount of time needed for pre-processing tasks. Although it has minor flaws, especially when dealing with a text with lot of inserted objects, it still can be of help and process correctly some part of documents, leaving the less for researchers to process by hand.

The next few steps of its development will be the implementation of the algorithm in a more user-friendly environment and its integration with existing language resources, such as electronic dictionaries, lexicons and grammars. These will provide an opportunity to NLP researchers to process PDF documents in a simplified manner, through user friendly GUI, but with more sophisticated level of classifying the text lines and recovering sentences. One such project is a *PDF corpora creator*, which is currently under development, as a part of this research.

Acknowledgements

This paper is the result of the research within the project 178006 financed by The Ministry of Science, Republic of Serbia.

References

Baumgartner, Robert., Sergio Flesca and Georg Gottlob. "Visual Web Information Extraction with Lixto". In *Proceedings of the 27th International Conference on Very Large Databases VLDB '01*, 119-128. San Francisco: Morgan Kaufmann Publishers, 2001.

Cunningham, Hamish, Diana Maynard, Kalina Bontcheva, and Valentin Tablan. "GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications." In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Philadelphia, 2002.

Kushmerick, N. "Wrapper induction: Efficiency and expressiveness." *Artificial Intelligence*, 118, 1-2 (2000): 15–68.

Liu, L., C. Pu and W. Han. "XWRAP : an XML-enabled Wrapper Construction System for Web Information Sources." In *Proceedings of 16th International Conference on Data Engineering*, 611–621. New Jersey: IEEE, 2000.

Muslea, Ion, Steve Minton and Craig Knoblock. "A hierarchical approach to wrapper induction." In Proceedings of the third annual conference on Autonomous Agents: Agents'99, 190–197. Seattle: ACM Press, 1999.

Paumier, S. "Unitex 2.1 User Manual". *Université de Marne-la-Vallée,* http://www-igm.univ-mlv.fr/~unitex/UnitexManual2.1.pdf

Silberztein, Max. "NooJ manual". Available at http://www.nooj4nlp.net/