Comparison of literary texts using biological sequence comparisons and structured document capabilities

Jacques André, Annie Morin and Hélène Richy

Irisa, Campus de Beaulieu, F-35042 Rennes cedex, France

Abstract—**Genetic Criticism requires new tools for computer aided text comparisons. This paper shows how string to string comparison algorithms used for biological sequence comparisons (and mainly the Smith and Waterman algorithm) may be applied to literary texts and how very large texts can be processed using the mathematical library MatLab. Futhermore, an environment for comparing strings based on structured documents principles is proposed. Gospel concordances are analyzed as examples.**

I. Introduction

Over the past couple of years, philologists, linguists and other Humanities experts have no longer needed to use large main frames for their text-based research. This is mainly due to new concepts such as the Internet (and the dissemination of large text bases), the standardization of structured texts (e.g. the *Text Encoding Initiative* [6]), the concept of active documents, ergonomics, etc.

However such research need new tools, in particular in the area of genetic critiscism. This is a relatively new field that studies the way authors think when writing a new work [10, 23]. Such a field needs to study manuscripts as images as well as texts. Studying a set of drafts also requires the study of the time dependent evolution of different versions (of manuscripts or printed texts). Text to text comparison algorithms are required. Note that closed domains such as authentication of genuine documents (either texts, scores, patents, etc.) have to use such comparison algorithms.

II. Text comparison

A. Goals of text comparison

1) First examples: Although we are working on specific French texts (such as Celine drafts, Proust manuscripts or medieval song-books [2]), examples in this paper will be taken from a single text: *The Gospel*, part of the New Testament from the Bible. It is a set of four documents (written by the Evangelists: Matthew, Mark, Luke, and John) telling the same story (the life of Jesus). Many translations exist in English, Arabic, Swahili, etc. as well as in alphabetical oriental languages and even in CJK (Chinese-Japanese-Korean). Today, most of these texts are on the Web[1] [12].

For example, here are the four versions of the same historical event according to the four authors.

> **Matthew, 26:34** "I tell you the truth," Jesus answered, "this very night, before the rooster crows you will disown me three times."
> **Mark 14:30** "I tell you the truth," Jesus answered, "today–yes, tonight–before the rooster crows twice you yourself will disown me three times."
> **Luke 22:34** Jesus answered, "I tell you, Peter, before the rooster crows today, you will deny three times that you know me."
> **John 13:38** Then Jesus answered, "Will you really lay down your life for me? I tell you the truth, before the rooster crows, you will disown me three times!

At the words micro-level, these 4 versions exhibit some differences concerning the vocabulary (*disown* versus *deny that you know me*), permutations (*I tell you the truth Jesus answered* versus *Jesus answered I tell you*), insertions/deletions (*before the rooster crows twice you yourself will* versus *before the rooster crows you will*), etc.

Differences may also be found at a macro-level, namely the order of the verses are not the same in all the versions. For example, these 4 items belong respectively to chapter 26(Matthew), 14 (Mark), 22 (Luke) and 13 (John). Futhermore, some continuous verses in one version are

[1] More precisely, texts quoted in this paper come from *The Bible Gateway*: `http://www.gospelcom.net/bible`.

found in various chapters in another (e.g. the sequence Matthew 6:22-34 corresponds to Luke 11:34-36 + 16:13 + 12:22-31; see Fig. 4 below).

In fact, these verses come from one version of the Gospel known as NIV. Its origin seems to be complex: initially there was one (known as Q, from the initial of the German word *Quelle*: spring, source), two, or four initial documents written in Arameen then translated to Greek on which the different versions according to each Evangelist were based. Futhermore, different versions or manuscripts exist for each text (known in English as NIV, RSV, KJV, DBY, YLT, etc.). For example, the verse from Matthew 26:34 is as follows according to these versions:

> **NIV** "I tell you the truth," Jesus answered, "this very night, before the rooster crows, you will disown me three times."
> **RSV** Jesus said to him, "Truly, I say to you, this very night, before the cock crows, you will deny me three times."
> **KJV** Jesus said unto him, Verily I say unto thee, That this night, before the cock crow, thou shalt deny me thrice.
> **DBY** Jesus said to him, Verily I say to thee, that during this night, before [the] cock shall crow, thou shalt deny me thrice.
> **YLT** Jesus said to him, 'Verily I say to thee, that, this night, before cock-crowing, thrice thou wilt deny me.'

These concordances have been studied, by hand, for centuries and synoptic presentations of the Gospel exist, such as [19, 4].

2) Expected results: Our goal is to offer Humanities researchers a toolbox to compare literary texts (such as Joyce's manuscripts or the different versions of the Gospel). Text comparison must be able to:

- Detect concording, i.e. similar parts of the texts.
- Produce a set of instructions to move from an original text $W1$ to the second version text $W2$ (like the Unix DIFF does, but sentence by sentence).
- Allow the philologist to exhibit differences interactively.
- etc.

Note however that comparing two texts is an undeterminate process! For example, it is impossible to say if the change from "After coming, Judas immediately went..." to "Immediately Judas went ..." is efected by:

- either adding "After coming", permuting "Immediately Judas" to "Judas immediately", keeping "went";
- or replacing "Immediately" by "After", adding "coming", keeping "Judas", adding "immediately", keeping "went";
- or changing "Immediately Judas" to "After coming Judas immediately", keeping "went";
- etc.

Hypotheses may be determined, such as "take the largest common sub-string first" or "prefer a solution upgrading $W1$ to $W2$ rather than the other way", etc. However, philologists have some knowledge or understanding of the texts they study and can help. That means that a tool comparing texts must offer researchers interactive dialogue facilities. See section D. below.

3) Word to word comparison: Many string comparison, sorting or searching algorithms have been written (see e.g. [1, 11, 13, 22] for surveys). A lot of applications use them for comparing files of characters, representing speech, sounds, or genetic sequences. These methods can easily be adapted to compare words, or groups of words, instead of characters. The application of such methods to compare text documents involves two new features:

1. Defining the **granularity** of the entities compared. For instance, when comparing word by word, some characters (punctuations) may be ignored, and identified as word separators. This is a syntactic concept.
2. Defining the **similarity** function of these entities. This similarity may be defined by a boolean function expressing that words are 'equal' or 'different'. The equality of two words is considered
   - either as a spelling equality, character by character, considering case or not,
   - or as a semantic equality, while considering that some words are equal if they belong to the same category, i.e. lexical, linguistic, etc.

   This is a semantic concept.

B. Brute Force Algorithm

An intuitive approach to our problem is simply to build a matrix showing what happens to each word of the original text $W1$ (a word may be kept in the new/compared text $W2$, modified, deleted or preceeded by new words, etc.).

Unformally, the algorithm works as follows. Let $W1$ and $W2$ be the two texts[2] (respectively with m and n words) to be compared. For example,

> **W1 (Matthew 26:49) =** Immediately Judas went to Jesus and said, "Hail, Rabbi!" and kissed Him.
> **W2 (Mark 14:45) =** After coming, Judas immediately went to Him, saying, "Rabbi!" and kissed Him.

[2] A state finite automaton analyses strings and builds vectors $W1$ and $W2$ where $W1_i$ and $W2_j$ are words.

	after	coming	judas	immediately	went	to	him	saying	rabbi	and	kissed	him
immediately				1								
judas			1									
went					1							
to						2						
jesus												
and										1		
said												
hail												
rabbi									1			
and										2		
kissed											3	
him							1					4

Fig. 1. Brute matrix from Matthew 26:49 (vertically) and Mark 14:45 (horizontally).

1. Build a matrix $M[1:m, 1:n]$ such that $M_{i,j} = 0$, except:

$$\begin{cases} M_{1,j} = 1 & \text{if} \quad W1_1 = W2_j \\ M_{i,1} = 1 & \text{if} \quad W1_i = W2_1 \\ M_{i,j} = M_{i-1,j-1} + 1 & \text{if} \quad W1_i = W2_j \end{cases}$$

   here $=$ having the meaning of spelling equality (see section 3) above).
   Fig. 1 shows this matrix for $W1$ and $W2$ texts above.
2. Look at the highest number (here it is 4) and mark the corresponding diagonal (from 1 to 4 here) with $\times$ signs.
3. Recursively, in the upper and lower diagonal submatrices again mark the largest diagonal items with $\times$ signs.
4. Mark the remaining diagonal submatrices with ∇ signs (delete) in the left column and $\triangleright$ signs (add) on the bottom line (note that here there is no replace sign for it is equal to ∇ then $\triangleright$ signs).
5. Delete any remaining number (considered as noise; however extra diagonals could help to determine repetitions, disseminations, etc.).

Fig. 2 shows the resulting matrix in our example. It exhibits a diagonal "path" explaining one process, out of several alternatives (see section D. below) to be used to move from text $W1$ to text $W2$. It may be read as follows (from the upper left corner to the lower right one):

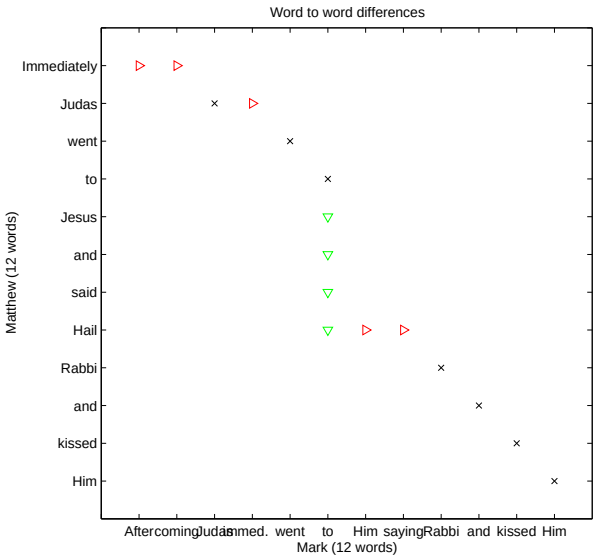

Fig. 2. Path matrix from brute matrix (see Fig. 1).

0	7	6	5	4	3	2	1	0	0
10	9	8	7	6	5	4	3	2	1
9	8	19	18	17	16	15	14	13	12
8	7	18	29	28	27	26	25	24	23
7	6	17	28	27	26	25	24	23	22
6	5	16	27	26	25	24	35	34	33
5	4	15	26	25	24	23	34	33	32
4	3	14	25	24	23	22	33	32	31
3	2	13	24	23	22	33	32	31	30
2	1	12	23	22	21	32	43	42	41
1	0	11	22	21	20	31	42	53	52
0	0	10	21	32	31	30	41	52	63

Fig. 3. Similarity matrix (Matthew 26:49 vs Mark 14:45) – compare with Fig. 1 and Fig. 2. N.B. here, columns 1 and 2 are missing.

- ∇ delete "immediately"
- $\triangleright$ add "after coming"
- $\times$ keep "Judas"
- $\triangleright$ add "immediately"
- $\times$ keep "went to"
- ∇ delete "Jesus and said Hail"
- $\triangleright$ add "him saying"
- $\times$ keep "Rabbi and kissed Him"

Heuristics may be used as well. For example, it is easy to detect permutations (thanks to symetrical diagonals), here the (1 word length) diagonals "Judas" and "immediately" are symetrical.

However this algorithm is not fully satisfactory for it is too dependent on micro-differences: if an extra word occurs in text $W2$ or if a single word is modified in the midst of say 30 words, then the comparison will find 2 common sequences instead of one. On the other hand, there is no way of considering the deletion/insertion of a sequence of words as being significant or not. This is why another more tolerant algorithm was used.

C. The Smith & Waterman Algorithm

Some string comparison methods have been adapted to search identities, or similarities, specially for biological sequence comparison; see [9, 16, 21, 25] for example. In order to identify common sequences, these methods calculate a "distance" between two strings. This distance is measured by considering the basic edit operations for transforming one string into another: substitution of characters, and insertion or omission of characters.

So, the distance between two texts is achieved while computing a $m \times n$ matrix, where m and n are the numbers of words in these texts. This matrix is called the **similarity matrix**. This method supposes that a **word substitution cost** *sbt* is defined for all words.

This matrix is computed recursively by the following equation (for $0 < i \leq m, 0 < j \leq n$):

$$H_{i,j} = \max\{E_{i,j}, F_{i,j}, H_{i-1,j-1} + sbt(W1_i, W2_j), \delta\}$$

where

$$E_{i,j} = \max \left\{ \begin{array}{l} H_{i,j-1} - \alpha \\ E_{i,j-1} - \beta \end{array} \right. \qquad F_{i,j} = \max \left\{ \begin{array}{l} H_{i-1,j} - \alpha \\ F_{i-1,j} - \beta \end{array} \right.$$

and

$$H_{i,0} = E_{i,0} = h_i \qquad H_{0,j} = F_{0,j} = v_j$$

A convenient setting of parameters α, β, δ, h_i and v_j leads to identify similar fragments. A backtrack procedure [26] determines similar fragments while considering local maxima in this matrix. The "degree of similarity" depends on the cost of insertions or omissions (α for the first, β for the next), thus allowing fragments of different sizes to be considered as similar fragments if multiple insertions or omissions are supported with a low cost.

Let us for instance turn back to the two verses Matthew 26:49 and Mark 14:45 (see section B. above). When locating similar fragments, the Smith-Waterman algorithm is computed with the following parameters:

- the cost of the first insertion or omission is 1 (α)
- the cost of the following insertion or omission is 1 (β)
- the word substitution cost between two words *w1* and *w2* is defined as:
  if $w1 = w2$ then $sbt(w1, w2) = 10$ otherwise $sbt(w1, w2) = -9$
- $h_i = v_i = 0 \ \forall \ i$
- $\delta = 0$

and produces the similarity matrix shown in Fig. 3.

The similar fragments producing the best scores in this matrix may be graphically displayed as in Fig. 2 (see also Fig. 4 produced, through Matlab [15], for larger texts).

In conclusion, the Smith-Waterman algorithm allows efficient detection of similar fragments within texts when setting relevant parameters. Note that if we compare Fig. 1 and Fig. 3, it can be seen that in the first one there are very small fragments while the second one offers a single fragment. The recent experimentations for speeding up the algorithms involved in biological sequence comparison [14] prompted us to experiment such algorithms in comparing very large texts. Comparing two versions of a 100 page novel (each page having 40 lines of 10 words) would require a $(100 \times 40 \times 10)^2$ i.e. a 16×10^8 element matrix (however this is a sparse matrix).

D. Experimentation and first results

The Smith-Waterman algorithm has been successfully adapted to genuine French literary texts [2] and to Gospel versions as well. Fig. 4 and Fig. 5 exhibit a larger text. The first results show some points which need improvement:

- Values to be affected to α, β, h_i, v_i, etc. are still not very manageable and more studies have to be done on the way to offer Humanities experts rules to be applied depending on the string lengths, on the expected tolerance, etc.
- Our examples show a single solution; however, some alternatives may be proposed. The question is "How to offer researchers a way to choose what they think is the good solution or how to propose an alternative?".
- How to establish the concordance of two texts? and of n texts?
- How to insert lexical analysis, dictionnaries, or semantic analysers into this scheme in such a way to eliminate trivial solutions?
- etc.

III. Text comparing environment

When large texts are concerned, the graphical presentation as shown in Fig. 4 is not sufficient. A more precise presentation of the compared texts is required: for instance, parallel visualization of the similar passages, hypertext linking similar fragments from the original texts, a synthetic document drawing together the similar fragments, a list of common fragments, etc. On the other hand, literary texts are no longersimple strings of words. Hypertext links or structures are more and more frequently involved in Humanities research, so new tools have to take this into account.

We propose therefore suggest using an editing environment to compare texts, that allows not only text editing, selecting passages for comparison, but also parametrizing the algorithm and producing an hypertext as a synthetic

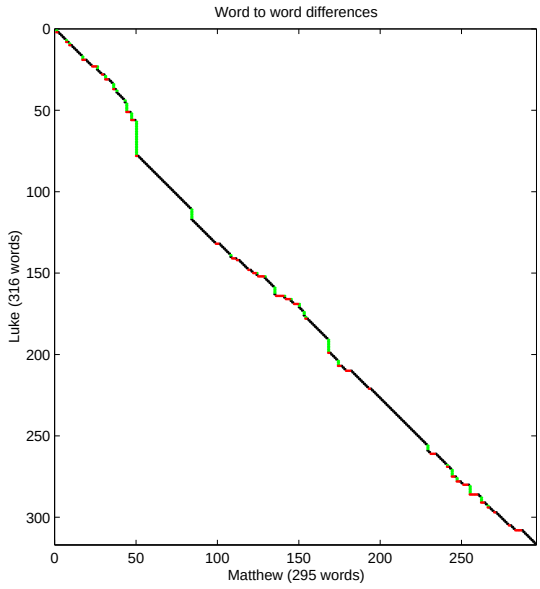

Fig. 4. Comparing Matthew 6:22-34 to Luke 11:34-36, 16:13 and 12:22-31; path after Smith & Waterman algorithm results.

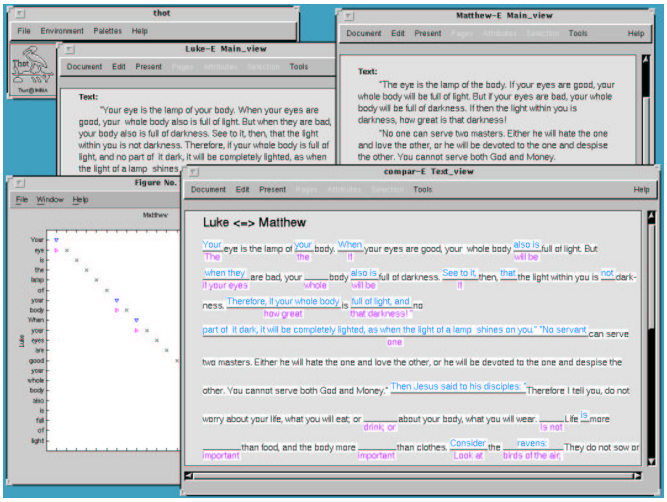

Fig. 5. Work window: on top, the two texts $W1$ (Luke 11:34-36+16:13+12:22-31) and $W2$ (Matthew 6:22-34); bottom left: the similarity matrix (detail); bottom right: the comparison view.

document. This editing environment provides the following features:

- original document and text content display
- interactive selection of relevant passages to be compared
- a parametrized comparison algorithm
- synthetic and hypertextual presentation of the compared texts

Our approach consists in integrating a text comparison algorithm as a new tool within an existing editing environment. This interactive editor provides a comprehensive environment for handling structured documents, and so produces various graphical aspects of the document, as described in the following sections. In particular, it is easy to add control menus allowing process calls (such as string comparison): the document is active [18].

A. Structured documents

Our comparing environment is built from an existing structured editor, called Thot, a recent evolution of Grif [17, 8]. All the editing functions of this structured editor are available to edit the original texts or the synthetic document. As this editor is structure-driven, a document is considered as a logical structure that organizes typed elements with attributes. Both hierarchical and non-hierarchical links may be created between elements. But, a logical model specifies the structure of a type of document, and any document is consistent with a logical model.

For instance, we have defined a new type of document to present the result of the comparison as one synthetic document (see the logical structure model in Fig. 6).

This logical model is consistent with the DTD (*Document Type Definition*) defined by the TEI (*Text Encoding Initiative* [6]):

- each original text is included in a division (<div1>)
- each original text is divided into fragments (<seg>), which are defined by the comparison algorithm
- anchors (<anchor>) are inserted at specific places within these original texts to represent the insertion of missing fragments
- the synthesis of the compared texts is described as an ordered list of couple of links (<LinkGrp>): the target of the first link is either a fragment or an anchor within the first division, the target of the second link is either a fragment or an anchor within the second division.

An example of the synthetic document, according to the TEI, produced by comparing the two sentences of the previous example in section C. is shown in Fig. 7.

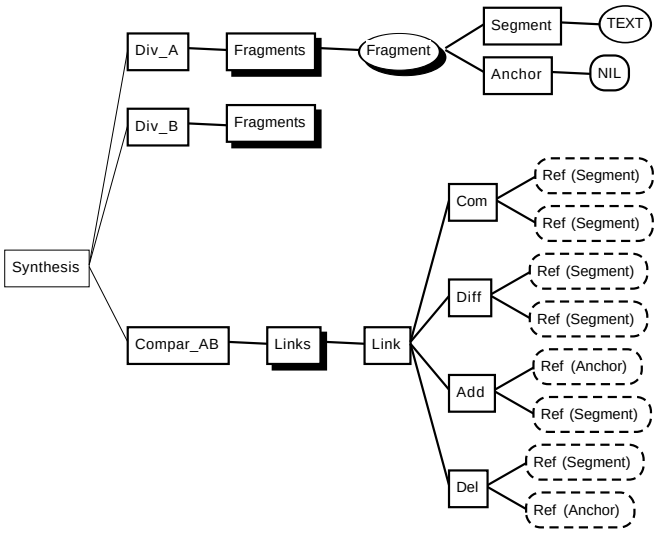

Fig. 6. Logical Model of the Synthesis Document.

```
<div1 id=TA corresp=TB>
<seg id=A1>When </seg> <anchor id=a1>
<seg id=A2>your eyes are good, your whole body </seg>
<seg id=A3>also is</seg>
<seg id=A4>full of light.</seg>
</div1>

<div1 id=TB corresp=TA>
<seg id=B1>If </seg>  <anchor id=b1>
<seg id=B2>your eyes are good, your whole body </seg>
<seg id=B3>will be</seg>
<seg id=B4>full of light.</seg>
</div1>

<linkGrp type='compar' domains='TA TB'>
 <link type='add' targType='anchor seg' targets='a1 B1'>
 <link type='del' targType='seg anchor' targets='A1 b1'>
 <link type='com' targType='seg seg' targets='A2 B2'>
 <link type='dif' targType='seg seg' targets='A3 B3'>
 <link type='com' targType='seg seg' targets='A4 B4'>
</linkGrp>
```

Fig. 7. Example of a (SGML/TEI) Synthesis Document

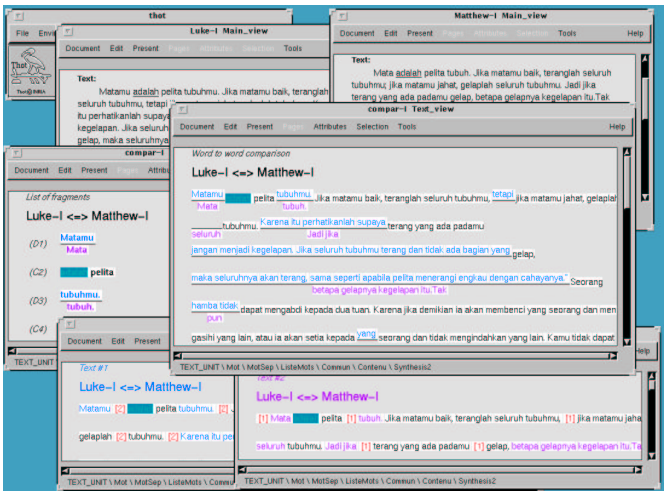

Fig. 8. Comparing the same texts as in Fig. 5 in Indonesian language

B. Views

In this context, it is possible to generically define the graphical aspect of the documents: presentation models contain all the necessary rules for producing the graphical aspect of all types of elements and attributes defined in a DTD. The presentation models also define different ways of presenting the same passages with different graphical aspects, within different windows (views), as shown in Fig. 5 and Fig. 8.

C. Multilinguism

Comparing texts in any language provokes two different classes of problems:

1. The comparison algorithm is quite independent of the language used: it works on byte strings. If one byte corresponds to one character or to half a character (as in Unicode [24]), the results are equivalent.
2. On the other hand, splitting texts into words, deleting the punctuation, taking care of semantic equality, etc. are language dependent and the text comparison environment (as described in section III.) has to be adapted accordingly.

Fig. 8 shows the result of comparing the same Gospel texts (in oriental languages) as those in English (Fig. 5).

IV. Conclusion

In this paper, we have first shown how character comparison algorithms can be adapted to compare large texts word to word. Various parametrized versions of this algorithm are currently studied in order to provide the users with various comparing functions: best, longest, nearest, similarity research, etc.

We have also described a comparison tool capable of producing a synthetic document by transforming two texts into an active hyperdocument. Our approach provides the Humanities experts with an experimental tool for examining the content of texts. Additional document styles can easily be implemented and we are studying various representations that can help in comparing two or more texts. As our document description uses or will use standards such as SGML/TEI, XML [5], DSSSL [7], the interchange of such electronic documents becomes easier. A new generation of software dedicated to literary texts will be soon available for Humanities researchers.

Acknowledgements: We would like to thank Dominique Lavenier for his advice, our students for their cooperation during the process of implementing this work, and Centre de Formation des Traducteurs, Terminologues et Rédacteurs (University of Rennes 2) for its proof reading.
This study was partly funded by a grant from the French "GIS Cognition".

V. References

[1] A. Aho, "Algorithms for finding pattern in strings", *Handbook of Theoritical Computer Science*, J. van Leeuwen ed., MIT Press, pp. 257–300, 1990.

[2] J. André and H. Richy, "Hypertextes ou documents structurés ? étude de cas en critique génétique", *Hypertextes et Hypermédias*, vol. 1, num. 2–4, pp. 13-26, September 1997

[3] L. Audoire, J.-J. Codani, D. Lavenier, P. Quinton, "Machines spécialisées pour la comparaison de séquences biologiques", *Technique et Science Informatique,* vol. 14, num. 1, January 1995.

[4] P. Benoit et M.-E. Boismard, *Synopse des Quatre Évangiles*, Les éditions du Cerf, Paris, 1990.

[5] T. Bray, C.M. Sperberg-McQueen, *Extensible Markup Language (XML),* Working Draft, 1997, http://www.w3.org/pub/WWW/TR, 1997.

[6] L. Burnard, and C.M. Sperberg-McQueen, *TEI Lite: An Introduction to Text Encoding, for Interchange*, http://www.uic.edu/org/tei/intros/, June 1995.

[7] ISO, *Information technology - Processing languages - Document Style Semantics and Specification Language (DSSSL),* vol. ISO/IEC DIS 10179, 1996.

[8] R. Furuta, V. Quint, J. André, "Interactively Editing Structured Documents", *Electronic Publishing,* vol. 1, num. 1, pp. 20-44, April 1988.

[9] O. Gotoh, "An improved Algorithm for Matching Biological Sequences", *J. Mol. Biol.,* vol. 162, pp. 705-708, 1982.

[10] A. Gresillon, *Éléments de critique génétique*, PUF, Paris, 1994.

[11] J. W. Hunt, T. G. Szymanski, "A fast algorithm for computing longuest common subsequences", *Comm. ACM,* vol. 20, num. 5, pp. 350-353, 1977.

[12] Institute for Christian Leadership, *A Guide to Christian Literature on the Internet*, June 1997 http://www.iclnet.org/pub/resources/christian-books.html

[13] D. E. Knuth, *The Art of Computer Programming, Volume 3: Sorting and Searching*, Addison-Wesley, Reading, MA, 1973.

[14] D. Lavenier, "Dedicated Hardware for Biological Sequence Comparison", *Journal of Universal Computer Science*, 2(2), February 1996.

[15] The MathWork, *MATLAB Documentation* High-performance Numeric Computation and Visualization Software (User's Guide, Reference Guide, etc.), The MathWorks, Inc. pub., Matick (Ma), 1992.

[16] S. Needleman, C. Wunsch, "A General Method Applicable to the Search of Similiraties in the Amino Acid Sequence of Two Proteins", *J. Biol. Mol.,* vol. 48, pp. 443-453, 1970.

[17] V. Quint, I. Vatton, "Grif, an Interactive System for Structured Document Manipulation", *Text Processing and Document Manipulation, Proceedings of the International Conference,* J. C. van Vliet, ed., pp. 200-213, Cambridge University Press, 1986.

[18] V. Quint, I. Vatton, "Making structured documents active", *Electronic Publishing - Origination, Dissemination and Design,* vol. 7, num. 2, pp. 55-74, June 1994.

[19] A. Resch, *Aussercanonische Paralleltexte zur den Evangelien*, Leipzig, 1893-1894.

[20] ISO, *Information processing - Text and office systems - Standard Generalized Markup Language (SGML),* vol. ISO 8879, October 1986.

[21] T.F. Smith and M.S. Waterman, "Identification of common molecular subsequences", *Journal of Molecular Biology*, vol. 147, pp. 195-197, 1981.

[22] G. A. Stephen, *String Searching Algorithms*, Lectures Notes Series on Computing - Vol. 3, World Scientific Publishing Co., Singapour, 1994.

[23] T. G. Tanselle, "Critical editions, Hypertexts and Genetic Critiscism", *Romanic Review*, vol. 86(3), pp. 581-595, May 1995.

[24] The Unicode Consortium, *The Unicode Standard – Worldwide Character Encoding*, Addison-Wesley, Reading Ma, 1991.

[25] M. Waterman, M. Eggert, "A New Algorithm for Best Subsequence Alignments with Application to tRNA-rRNA Comparisons", *J. Mol. Biol.,* vol. 197, pp. 723-728, 1987.

[26] M. Waterman, "Mathematical Methods for DNA Sequences", *CRC Press, Inc.,* 1989.