# Alignment of noisy unstructured text data

**Julien Bourdaillet** and **Jean-Gabriel Ganascia**
Université Pierre et Marie Curie
Laboratoire d'Informatique de Paris 6
Computer Science Department
julien.bourdaillet@lip6.fr, jean-gabriel.ganascia@lip6.fr

## Abstract

This paper describes a textual aligner named MEDITE whose specificity is the detection of moves. It was developed to solve a problem from *textual genetic criticism*, a humanities discipline that compares different versions of authors' texts in order to highlight invariants and differences between them. Our aligner handles this task and it is general enough to handle others. The algorithm, based on the edit distance with moves, aligns duplicated character blocks with an $A^*$ heuristic algorithm. We present an experimental evaluation of our algorithm by comparing it with similar ones in four experiments. The first one deals with the alignment of texts with a large amount of repetitions; we show it is a very difficult problem. Two other experiments are duplicate linkage and text reuse detection. Finally, the algorithm is tested with synthetic data.

## 1 Introduction

Noisy unstructured text data may have different sources such as the Web, handwritten texts or historical texts. We found such data when starting a collaboration with a team of philologists who created a new school of literary studies in the seventies called textual genetic criticism [Deppman *et al.*, 2004]. The goal of this discipline is to study the genesis of a text in order to understand the will and the work of an author. This is done by studying drafts let by authors during the writing process.

An author begins writing a first draft on a paper sheet. Afterwards, he modifies the text by annotating, barring words or correcting spelling faults, and inserting words or sentences. These modifications can be done several times on the same draft. At the end, these drafts can be very hard to read and understand, and cannot be studied directly by machine via optical character recognition (OCR). From these drafts, philologists extract linear transcriptions called versions. Version extraction from drafts is out of machine capabilities and need to be human-made. For example, in the short story entitled "La Robe Noire" by Andrée Chedid (an Egyptian author writing in French), there are seven versions on the same draft.

The pairwise comparisons of these versions are done by philologists. They look for four kinds of modifications between two versions: insertions, deletions, replacements and moves which are the four operators defined by textual geneticians. This is a very time-consuming task and the analysis of long texts (e.g. an entire novel) can require a huge amount of time. This is the reason why we created a software application, named MEDITE, that automates this process.

The task of textual alignment can be summarized as follows: being given two text versions, invariants and differences between them have to be identified. The input data is in the form of unstructured raw text files; they do not need to be structured as XML files or database records. There are two types of duplicated character blocks: invariant blocks, conserved at the same positions between the two texts, and moved blocks, shifted from one position in the first text to another one in the second text. The goal is to identify and link these duplicated blocks. When there are a lot of differences between the two texts, i.e. a lot of noise, the duplicate alignment becomes a difficult task.

We present related literature in Section 2. We modelize textual alignment with the notion of edit distance with moves, as presented in Section 3; this formalism handles exactly the operators defined by textual geneticians. Further, the move detection allows the identification and linkage of more duplicated data than simpler algorithms based on edit distance or longest common subsequence. The computation of the edit distance with moves is exponential and we developed a heuristic algorithm implemented inside MEDITE. Our algorithm is character-based, and not only word- or sentence-based; this is necessary for textual genetic criticism where even small modifications have to be identified.

In Section 4, four experiments are presented. In a previous work, MEDITE has been evaluated with literary texts provided by textual geneticians [Bourdaillet and Ganascia, 2006]. In this paper we present new results for several tasks. In literary texts, there are usually very few repetitions because authors avoid them; whereas this is not a problem in scientific texts and often even a necessity. While studying these texts with a large amount of repetitions, we found that it was a difficult problem for textual aligners. The first experiment compares our algorithm with similar ones for scientific text alignment, we show that this kind of texts present specificities. The second experiment deals with duplicate detection and linkage on two datasets: the "Restaurant" dataset and a dataset of our own, the "Conference" dataset; because our algorithm discovers moves between texts, it is able to link duplicates whereas standard aligners do not. The third experiment is the detection of text reuse between texts sharing short passages, this is also a consequence of move detection capability. And the fourth experiment is synthetic data alignment when a reference alignment is known. Finally, we conclude in Section 5.

## 2 Background

In computer science, textual alignment is viewed as a particular case of sequence alignment. Most of the algorithms designed to solve this problem are optimisation procedures by dynamic programming. Levenshtein introduced the notion of edit distance [1966] which enables the transformation of a sequence into another one with three operators: insertions, deletions and replacements. This notion has been extensively studied and developed; refer to [Bergroth *et al.*, 2000] for a recent survey. The extension of the notion of edit distance with block moves was introduced by [Tichy, 1984]. [Lopresti and Tomkins, 1997] introduced several models of block edit distance. [Shapira and Storer, 2002] proved the NP-completeness of the computation of the edit distance with moves between two sequences and gave worst-case bounds for a greedy approximation algorithm of this problem.

Sequence alignment has also been widely studied in bioinformatics [Gusfield, 1997]. There are two classes of alignment algorithms: global aligners issued from [Needleman and Wunsch, 1970] fully align two sequences by minimizing an edit distance whereas local aligners issued from [Smith and Waterman, 1981] find regions of high similarity between two sequences. [Wilbur and Lipman, 1984] introduced the notion of fragment alignment: high similarity fragments are chained together to form a full alignment. This is now widely used by recent aligners such as AVID [Bray *et al.*, 2003].

In biological sequences there are a lot of duplicated subsequences: half of the human genome is estimated to be duplicated subsequences. These duplications are a problem for biologists and compared sequences are generally preprocessed with tools that mask duplications such as RepeatMasker [Smit *et al.*, 1996].

Machine translation uses alignment, but this is a bilingual alignment (whereas our task is unilingual alignment) and the granularity is most of the time sentence-based. Word-based alignments are still in an early research stage [Chiao *et al.*, 2006]. Further, the compared texts are known to be the translation of one to the other and this implies that there is generally no noise between texts.

Textual alignment software applications are generally issued from the source code alignment community whose *Diff* [Myers, 1986] is the most famous example. For this task, line-by-line comparison is sufficient but for our task, this granularity is too large. Some of the commercial applications for textual alignment present acceptable results [Bourdaillet and Ganascia, 2006]; we compared them with MEDITE in our experiments (Section 4). The Greedy String-Tiling algorithm [Wise, 1996] is close to our algorithm, it handles block moves by exact matching of maximal length strings. TESAS uses an interesting approach by comparing and scoring all the sentences together [Clough *et al.*, 2002]. We also compared MEDITE with these last two algorithms in Section 4.

[Feng and Manmatha, 2006] present a hierarchical HMM-based algorithm that aligns the OCR output of book with another edition of the same book. Hapax (words present only once in each text) are first aligned; then word sequences between hapax are aligned with a HMM model; and finally text between exactly matched words is aligned at character level. We compare MEDITE to this algorithm in an experiment of synthetic data alignment in Section 4.

A textual alignment for paraphrase extraction from a unilingual corpus is presented in [Regina Barzilay, 2001]. Sequences such as two different translations of a same novel are aligned to find paraphrases. The algorithm relies on the following steps: first sentences are aligned; then the context between matching words inside sentences is studied; context features are extracted (mainly morpho-syntaxic tags); a machine learning algorithm keeps only "good" contexts which correspond to paraphrases.

## 3 Methods

The alignment task issued from textual genetic criticism can be modelized by the edit distance with moves. Two character sequences $A$ and $B$ are defined over a finite alphabet $\Sigma = \{a, ..., z\} \bigcup \{A, ..., Z\} \bigcup \{\mathcal{ACC}\} \bigcup \{\mathcal{SEP}\}$ where $\{\mathcal{ACC}\}$ is the set of accentuated and diacritic characters and $\{\mathcal{SEP}\}$ is the set of separators, mainly punctuation marks.

Four operators are given: character insertion, deletion and substitution and block moves. A block is a 3-tuple $(p, q, l)$ where $p$ is the position in $A$, $q$ the position in $B$ and $l$ the length of the block. A cost is assigned to each operator. The goal is to find a sequence of operations of minimum cost which transforms $A$ into $B$. Character blocks not involved in an edit operation are called invariant blocks: they are present in $A$ and $B$. The decomposition of $A$ and $B$ into a list of inserted, deleted, substituted, moved and invariant blocks forms an alignment.

The computation of the edit distance with moves between two sequences is NP-complete [Shapira and Storer, 2002], that is why we developed a heuristic algorithm to address this problem.

Our algorithm is closely related to fragment alignment algorithms sketched in Section 2. Sequences are processed in four steps. The first step identifies repeated character blocks between $A$ and $B$. The second step aligns these repeated blocks in order to determine which are invariant and which are moved. The third step is a recursive iteration of steps 1 and 2 between every pair of aligned blocks during the step 2. The last step is the deduction of insertions, deletions and replacements. Figure 1 presents the algorithm.

**Repeated block identification** The identification of repeated character blocks is done by building a generalized suffix tree between $A$ and $B$ [Ukkonen, 1995]. This data structure enables to find all repeated character blocks between $A$ and $B$. The size of this set of blocks is exponential and only

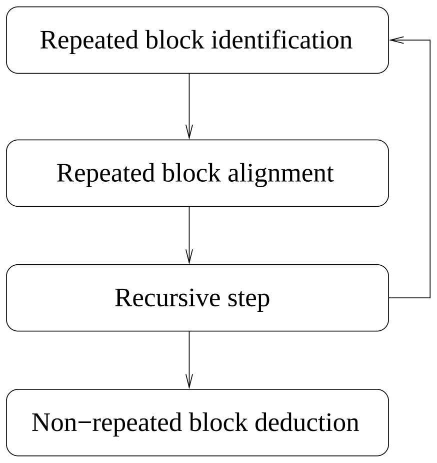

Figure 1: MEDITE's algorithm

a subset is interesting: the subset of super maximal exact matches (SMEMs) [Gusfield, 1997]. A match $(p, q, l)$ is a SMEM if and only if:

- $A[p..p + l] = B[q..q + l]$ (exact matching);
- $A[p - 1] \neq B[q - 1]$ and $A[p + l + 1] \neq B[q + l + 1]$ (maximality);
- and neither $A[p..p + l]$ nor $B[q..q + l]$ are included in another maximal match (super-maximality).

This definition does not prevent overlapping between SMEMs whereas SMEMs have to be non-overlapping in our application. Overlaps are resolved heuristically by cutting them on separators because it is better to have an inter-word cut than an intra-word cut in natural language sequences. This first step results in two lists $A'$ and $B'$ of non-overlapping SMEMs.

To illustrate the algorithm, we use the following example: the alignment of the two sentences "This morning the cat observed little birds in the trees." and "The cat was observing birds in the little trees this morning, it observed birds for two hours.". We use MEDITE's capability to compare sequences case- and separator-insensitively to treat this example. After the first step, the following SMEMs are identified: "$\boxed{\text{This morning}}$ $\boxed{\text{the cat}}$ $\boxed{\text{observed}}$ $\boxed{\text{little}}$ $\boxed{\text{birds in the}}$ $\boxed{\text{trees}}$." and "$\boxed{\text{The cat}}$ was observing $\boxed{\text{birds in the}}$ $\boxed{\text{little}}$ $\boxed{\text{trees}}$ $\boxed{\text{this morning}}$, it $\boxed{\text{observed}}$ birds for two hours.". The word "birds" repeated 3 times does not appear in the SMEM list because the first two occurrences are included in longer SMEMs. Had it been in the list, the super-maximality would not be respected.

**Repeated block alignment** Each of these SMEMs can be either an invariant or a moved block. The pairwise alignment of SMEMs enables to make the decision: aligned SMEMs are considered as invariants and unaligned as moved. Because the space of possible alignments is combinatorial [Kolman *et al.*, 2004], we use an A$^*$ heuristic algorithm.

Possible alignments are evaluated with a cost function $c$; the goal is to find a minimal cost alignment. This is equivalent to a shortest path problem where the goal state is the minimal cost alignment. The initial state corresponds to the state where no pairing has been chosen yet. At each step of the algorithm one pairing must be chosen; this is done by estimating the alignment cost induced by each possible pairing with the function $c$. It is a greedy best first search algorithm, so the best pairing is chosen and then this process is iterated until the goal is achieved. In order to find the goal state, the heuristic has to be admissible, i.e. to never overestimate the distance to the goal. In our case the heuristic must never overestimate the cost of an alignment; we detail below why $c$ is admissible.

The evaluation of the alignment cost induced by the pairing of $A'_i$ with $B'_j$ is computed with $c(i, j)$ which breaks down into the cost of the alignment so far $g(i, j)$ and the heuristic cost of the next blocks to align $h(i, j)$, such that $c(i, j) = g(i, j) + h(i, j)$. These costs are computed using the following formulas:

- $U(i, j) = unaligned(A'[1..i - 1], B'[1..j - 1])$ is the set of unaligned blocks during previous steps, those that have not been paired and considered as moved.
- $g(i, j) = \Sigma_{b \in U(i,j)} |b|$ is the sum of previously unaligned blocks' size, that is the alignment cost is charged by moved blocks only.
- $SD(i, j) = A'[i + 1..|A'|] \ominus B'[j + 1..|B'|]$ is the symmetric difference of the next blocks to align during next steps. Blocks in $SD(i, j)$ are present either only in $A'[i + 1..|A'|]$ or only in $B'[j + 1..|B'|]$. It will never be possible to pair them during next steps and because of that they will be considered as moves and will charge the alignment cost.
- $h(i, j) = \Sigma_{b \in SD(i,j)} |b|$ is the sum of these blocks' size, i.e. $h(i, j)$ is the lower bound of the alignment cost of the next blocks. It never overestimates the alignment cost because the minimal alignment cost will be $h(i, j)$; this is why $c$ is admissible and A$^*$ finds the optimum.

This computation is equivalent to finding an alignment that is optimal in the sense of the maximization of the sum of invariant block size and the minimization of the sum of moved block size.

In our example, after the second step, the following aligned blocks (in bold) are considered as invariant; the other squared blocks are moved: "$\boxed{\text{This morning}}$ $\boxed{\textbf{the cat}}$ $\boxed{\text{observed}}$ $\boxed{\text{little}}$ $\boxed{\textbf{birds in the}}$ $\boxed{\textbf{trees}}$." and "$\boxed{\textbf{The cat}}$ was observing $\boxed{\textbf{birds in the}}$ $\boxed{\text{little}}$ $\boxed{\textbf{trees}}$ $\boxed{\text{this morning}}$, it $\boxed{\text{observed}}$ birds for two hours.".

**Recursive step** The third step consists in looping over the pairings resulting from step 2 and in considering the subsequences between each pair of aligned blocks. These subsequences are processed again with steps 1 and 2. It allows the pairing of new blocks which are then included in the main alignment. This recursive step enables the pairing of blocks which would otherwise have been left unaligned.

In the example, the subsequences “observed little” and “was observing” occur between the invariant blocks “the cat” and “birds in the”. Recursive step 1 finds the SMEM “observ” and recursive step 2 aligns them. Finally the final alignment becomes “$\boxed{\text{This morning}}$ $\boxed{\textbf{the cat}}$ $\boxed{\textbf{observ}}$ed $\boxed{\text{little}}$ $\boxed{\textbf{birds in the}}$ $\boxed{\textbf{trees}}$.” and “$\boxed{\textbf{The cat}}$ was $\boxed{\textbf{observ}}$ing $\boxed{\textbf{birds in the}}$ $\boxed{\text{little}}$ $\boxed{\textbf{trees}}$ $\boxed{\text{this morning}}$, it observed birds for two hours.”. The moved block “observed” is lost but the invariant block “observ” is discovered. The algorithm favours local similarities rather than long-distance matching.

**Other block deduction** Insertions, deletions and replacements can then be deduced. Deletions are non-repeated blocks in $A$ and insertions are non-repeated blocks in $B$. Further, when there is a deleted block $d$ in $A$ and an inserted block $i$ in $B$ between two pairs of aligned blocks, and the ratio $|d|/|i|$ reaches a threshold $t$, then $d$ and $i$ are transformed in replacements $r_1$ and $r_2$, meaning that $r_1$ has been replaced by $r_2$. $t$ is arbitrarily set to 0.5.

In the example, the not framed block of the first sequence is considered as a deletion and the three not framed blocks of the second sequence are considered as insertions.

## 4 Experiments

### 4.1 Experiment 1: very noisy text alignment

This first experiment processes very noisy data which have been provided by epistemologists. Claude Bernard was a French physiologist of the nineteenth century, one of the fathers of modern medicine. During some experiments he was studying the effects of curare (poison) on living organisms by injecting curare into frogs. Experiment notes were taken by an assistant during the experiment; these notes are short sentences written in telegraphic style. Some years later, Claude Bernard wrote an academic synthesis exposing his scientific method where he explained again these experiments by rewriting the text from the notes. This second text is a scientific paper, all sentences are grammatically correct and they can be very different from those of the notes.

The notes and the synthesis are closely related but the form is very different between these two texts: this is a first notion of noise. And there is a large amount of duplicate words between the two texts: this is a second notion of noise.

The first text (experiment notes) is 5090 characters long for 884 words and the second text (academic synthesis) is 8559 characters long for 1525 words. Table 1 presents the occurrence number of the top ten words. Words such as determiners and articles (i.e. function words) have been omitted from this count. For example, it can be seen that the word “patte” (“leg”) is repeated 58 times and covers 2.18 % text size. Word frequencies clearly follow a Zipf distribution (the frequency of words is inversely proportional to their rank in the frequency table) but in these texts the weight of most frequent words is particularly important. The top curve of Figure 2 presents the cumulated percentage of Claude Bernard text size in function of word frequencies. The bottom curve was established from a corpus of 8 short stories and essays (of size similar to Claude Bernard’s texts); it is the average cumulated percentage text size of these 8 texts. For example, the 15 most frequent words of Claude Bernard’s texts represent 17 % of texts’ size whereas, on average, they represent 8.3 % of the size of corpus texts. 47.68 % of the two text size consists in words repeated at least 3 times; this involves ambiguity for the linkage of these words.

| Word | # Occ. | % Texts |
|---|---|---|
| patte | 58 | 2.18 |
| dans | 57 | 3.89 |
| pince | 29 | 4.99 |
| plus | 27 | 5.80 |
| grenouille | 27 | 7.83 |
| quand | 25 | 8.77 |
| mouvements | 24 | 10.57 |
| liée | 24 | 11.29 |
| muscles | 23 | 12.51 |
| sont | 21 | 13.14 |

Table 1: Duplicate words in Claude Bernard’s texts. “# Occ.” is the number of word occurrences in both texts. “% Texts” is the cumulative percentage of word size (in characters) in both texts.

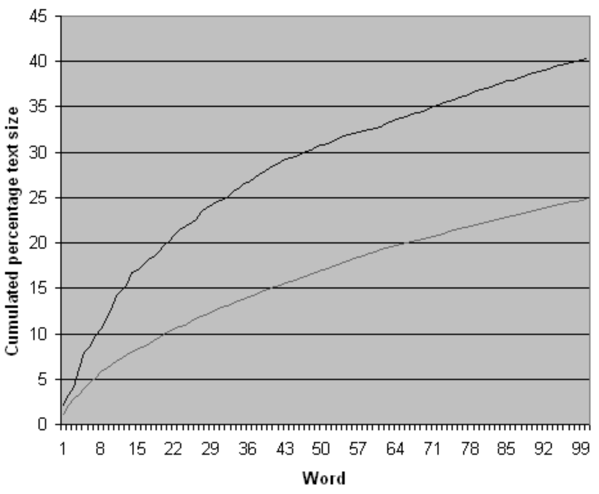

Figure 2: Cumulated percentage text size of words in Claude Bernard’s texts (top curve) and in the corpus (bottom curve)

In this experiment, Claude Bernard’s texts are compared with MEDITE, Microsoft Word (using the ‘Document comparison and fusion’ command), SoftInterface DiffDoc and TESAS. In previous work [Bourdaillet and Ganascia, 2006], we compared MEDITE with ten other alignment software and only Word and DiffDoc gave acceptable results; the other ones gave poor results. TESAS was not considered in our previous work.

The large amount of repetitions and noise between the texts and the granularity of the alignment (finer than sentence-based) avoid the definition of a reference alignment because a lot of choices are possible; and consequently Information Retrieval’s measures, like precision and recall, can not be used.

| | MEDITE | Word | DiffDoc | TESAS |
|---|---|---|---|---|
| # invariant characters | 3876 | 2386 | 2918 | 1706 |
| # invariant blocks | 256 | 162 | 356 | 120 |
| # moved characters | 1205 | 0 | 0 | 3152 |
| # moved blocks | 105 | 0 | 0 | 434 |
| # duplicated characters | 5081 | 2386 | 2918 | 4858 |
| # duplicated blocks | 361 | 162 | 356 | 554 |
| % duplicated / \|texts\| | 37.23 | 17.48 | 21.38 | 35.59 |

Table 2: Comparison of the analysis of Claude Bernard's texts with MEDITE, Microsoft Word, SoftInterface DiffDoc and TESAS.

| | MEDITE | Word | DiffDoc | TESAS |
|---|---|---|---|---|
| # invariant characters | 3628 | 2300 | 2662 | 1706 |
| # invariant blocks | 218 | 157 | 280 | 120 |
| # moved characters | 703 | 0 | 0 | 1208 |
| # moved blocks | 45 | 0 | 0 | 94 |
| # duplicated characters | 4331 | 2300 | 2662 | 1457 |
| # duplicated blocks | 263 | 157 | 280 | 107 |
| % duplicated / \|texts\| | 31.73 | 16.85 | 19.50 | 21.35 |

Table 3: Same results as Table 2 after manual analysis of alignments.

The results of this comparison are presented in Table 2. The four applications identify invariant character blocks between the two texts but only MEDITE and TESAS identify moved blocks. Duplicated blocks are defined as the sum of invariant and moved blocks. MEDITE identifies 361 duplicated character blocks representing 37.23 % of the size of the two texts, Microsoft Word identifies 162 duplicated blocks (17.48 %), SoftInterface DiffDoc 356 blocks (21.38 %) and TESAS 554 blocks (35.59 %).

These results are raw results. We manually checked the quality of alignments produced to evaluate if each block pairing was correct. The detailed analysis of MEDITE's results shows that most of the invariant block pairings are correct, but some can be considered as *abusive* pairings: two blocks can be paired because they are actually identical but when examining manually (and semantically) the context, we can see that the context is different and the pairing abusive. It comes from the second notion of noise we introduced: the large amount of duplicate words between the two texts. Nevertheless, MEDITE identifies important pairs: those which are semantically important for the meaning of the sentence. Further, some moved blocks pairing are abusive: those of function words which have been removed. Table 3 presents these results where abusive pairings have been removed.

The detailed analysis of Word's results shows that Word only finds a subset of MEDITE's invariant block pairings and misses a lot of pairings. Some abusive pairings were also removed. The analysis of DiffDoc's results shows that it is the application finding the largest number of invariant block pairings, but actually a large part of these pairings is the pairing of function words: words which are not semantically important. Even after removing abusive pairings, it aligns the largest number of blocks, but a large part of these pairings are function word pairings which are correct in context but do not bring important information. Similar results occur with TESAS: it finds the largest number of moved blocks but a large part of these pairings are function words pairing that have been removed in the second table. TESAS finds more moved blocks than MEDITE but this is because TESAS does not align them correctly as invariant blocks whereas MEDITE does.

Word found 10 pairings that MEDITE did not find: 3 were semantically important and 7 were function words. DiffDoc found 16 pairings that MEDITE did not find: 2 were semantically important and 14 were function words.

Our results are better because of the recursive step of our algorithm. Without this step, what we call *masking effects* occur (see the example of "observe" in Section 3): long-distance large block pairings mask and prevent local smaller block pairings. This notion is close to the notion of *shadow effect* identified by [Arslan *et al.*, 2001] in biological sequences where short highly similar pairings are discarded because of longer less similar pairings.

Two other software applications have also been evaluated. WinMerge is one of the rare commercial aligners which detect moved blocks but results were very poor: it identified only one invariant block of 18 characters and no moved blocks. Reuse Analysis Workbench (RAW), an implementation of the Greedy String-Tiling algorithm by [Clough *et al.*, 2002], was also evaluated but the results were only the highlighting of duplicates, blocks were not paired together and the alignment evaluation was not possible.

For this experiment, we compared only Claude Bernard's texts; we are seeking similar texts with a large amount of duplicates in order to have more statistically significant results. But more generally the evaluation of the goodness of an alignment remains an open question. Because of the lack of reference alignment, this problem is similar to unsupervised learning (i.e. clustering) where classes would be block types. For clustering, measures such as homogeneity, compactness or similarity between objects inside classes, or separation between classes are used. To use these measures in our case, features would have to be extracted from blocks such as length, position, character distribution or part-of-speech tags. We are currently evaluating which features to extract in order to continue this work.

## 4.2 Experiment 2: duplicate linkage

The second experiment deals with duplicate detection inside collections. Two datasets are processed: the "Restaurant" dataset and the "Conference" dataset. For this experiment, only MEDITE and RAW are compared. Word and DiffDoc completely fail to align these datasets because they do not handle moves. TESAS is based on sentence alignment and, because of the large number of punctuation marks in the two datasets (due to abbreviations in the first dataset, and author lists, address lists and paper titles in the second dataset), the algorithm fails to pair duplicates.

**"Restaurant" dataset** The "Restaurant" dataset is a collection of 864 restaurant records from the Fodor's and Zagat's restaurant guides that contains 112 duplicates. MEDITE was not developed for this kind of duplicate identification but, because it detects moves, it is able to handle this task.

The comparison focuses on the text files of two restaurants lists where we previously deleted the phone numbers. Table 4 presents the results of the experiment. For the test 1, we compared the two texts by allowing the matching of full words only: SMEM overlapping cuts inside words were not permitted (see Section 3), i.e. all blocks are n-grams. For the test 2, these cuts were permitted and it enables the matching of longer character blocks.

| | Test 1 | Test 2 |
|---|---|---|
| # names only | 17 | 10 |
| # addresses only | 5 | 11 |
| # names + adresses | 84 | 88 |
| # missed | 6 | 3 |

Table 4: "Restaurant" dataset comparison. "# names only" is the number of duplicates where only the name matching is correct. "# addresses only" is the number of duplicates where only the address matching is correct. "# names + adresses" is the number of duplicates where both name and address matchings are correct. "# missed" is the number of duplicates where matching has not been found.

In the first test, only 6 duplicates are missed and in the second only 3. These good results are possible because of the moves' detection by MEDITE's algorithm: duplicated restaurants with different names occur at different positions in the two files and only an algorithm that detects moves can link them. Further, the matching between any identical subsequences, and not only between words, enables the pairing of duplicates where the name or the address changes.

**"Conference" dataset** The "Conference" dataset is composed of two collections: the list of scientific committee members at the LREC 2006 conference and the list of accepted papers for this conference. We copied these lists from the conference's website, resulting in two raw text files (without any HTML markers).

The duplicates are the 178 committee members who appear in the list of accepted papers. The list of committee members is 302 people long and each file line is in the format: Surname First Name, Laboratory and/or University, City and/or Country. The format is not strict, some commas can be omitted. The paper list is 540 papers long and each paper is in the format: Paper Number First Name Surname, Paper Title. Many papers have several authors; each field is separated by several carriage returns; in some papers there are author initials instead of full first name.

We compared the two lists with MEDITE (intra-word cuts were not permitted). 161 of the 178 duplicates were found. The missed duplicates are: in 5 cases, only the first name was found but not the surname; in 5 others, the first name was matched with the first name of another person; in 3 cases, the surname matching was between persons with the same name; and 2 duplicates were missed.

This duplicate linkage experiment shows that MEDITE presents good results. The algorithm was not developed for this task but it is general enough to handle it. MEDITE generates some noise by pairing cities in the first dataset and countries in the second, but this noise could be avoided with simple adaptation of the algorithm: regular expression on punctuation marks or city/country stopword preprocessing.

## 4.3 Experiment 3: text reuse

The third experiment concerns text reuse detection. This notion introduced by [Clough *et al.*, 2002] describes how content from a source is reused and eventually modified to create a new text. This is typically the case of plagiarism or of journalism where news agencies produce texts that are reused by newspapers.

We compared two pairs of such texts. The first test is the alignment of a French news agency's dispatch and a newspaper article derived from the dispatch where four paragraphs have been copied with some modifications inside. The second test is the alignment of two mid-seventeenth century English newspaper article from the Lancaster Newsbook Corpus where four paragraphs have also been copied with modifications. The goal is to link the related paragraphs.

| # paragraphs linked | MEDITE | Word | DiffDoc | TESAS |
|---|---|---|---|---|
| French texts | 4 | 0 | 4 | 4 |
| English texts | 4 | 0 | 0 | 4 |

Table 5: Results of the text reuse experiment

Table 5 presents the results of the experiment. Only MEDITE and TESAS passed the two tests. TESAS was especially designed for this task, whereas MEDITE was not but achieves the same results.

## 4.4 Experiment 4: synthetic data alignment

The goal of this experiment is to evaluate the quality of MEDITE on synthetic data when a reference alignment exists. Given a text and a noise model, a second text is generated by altering the first one; the alignment between the texts is recorded during the alteration process. Then, it is possible to evaluate the quality of the aligner by comparing its results with the reference alignment. This experiment is similar to the one presented in [Feng and Manmatha, 2006].

**First noise model** The noise model allows to generate a modified text from an original text in the following way. Ratios of insertions, deletions and replacements on the original text size are set before processing. Then character blocks are repeatedly inserted in the modified text, deleted in the original text and replaced between both texts until ratios are reached. The positions where operations occur are chosen randomly over the size of the texts (overlapping operations are not allowed). The operations deal with character blocks rather than single characters in order to simulate real operations on words; the size of character blocks is randomly chosen between 1 and 25 (i.e. single characters are allowed). During this process the positions in the original text where deletions occur, the positions in the modified text where insertions occur and the positions in both texts where replacements occur are recorded. Finally, a reference alignment between the original and the modified text is produced.

For the experiment we have chosen a 520K characters long novel for the original text. Five modified documents were generated with the noise model. Then the documents were aligned with MEDITE and accuracy rates evaluated. Two series of tests with different modification ratios were conducted: in the first one there are 5 % of insertions, 5 % of deletions and 5 % of replacements, which means that there is a 15 % difference between original and modified texts; in the second series, the ratio is set to 10 %, meaning that there is a 30 % difference. For each of the four kinds of characters (insertions, deletions, replacements and invariants) the accuracy rate is defined as the number of correctly aligned characters / the total number of characters. Then the average accuracy rate of these four rates is calculated. For the average weighted accuracy rate, the four accuracy rates are weighted with their ratio of the texts' sizes. The average run times of alignments is also calculated. Table 6 presents the results of this experiment.

| Modification ratio | 5 % | 10 % |
|---|---|---|
| Average accuracy rate | 94.48 % | 89.27 % |
| Average weighted accuracy rate | 98.16 % | 94.0 % |
| Average run time | 11 mn 5 s | 27 mn 53 s |

Table 6: Results of the synthetic data alignment with the first noise model

When comparing our results with those described in [Feng and Manmatha, 2006], we observe that our average accuracy rates are 5 points inferior (on average). The algorithm of Feng and Manmatha pre-aligns texts by looking for hapax; our algorithm could be easily modified with this pre-alignment step which would increase our results. On the other hand, our algorithm detects moves between texts whereas Feng and Manmatha's algorithm does not. Further, we think that weighted accuracy rates reflect better the quality of an alignment because without weighting the weights of modified characters are overweighted. The introduction of weights enables to balance accuracy rates.

The experiment was realized on a Pentium 4, 2.4 GHz with 1 GB of RAM. MEDITE is implemented in Python, a high-level language good for prototyping but slow. A C language implementation would allow to win an order of magnitude in speed. Nevertheless, the speed bottleneck of our algorithm is the calculation of symmetric differences between lists of SMEMs (see Section 3) which is quadratic in the length of the lists.

**Noise model with moves** This second noise model is similar to the previous one but moves are also included in the model to generate the modified text from the original. Hence character block moves, shifted from one position in the original text to another one in the modified, are introduced. Again modification ratios of 5 and 10 % for each operation are tested resulting in texts with a 20 and a 40 % difference respectively. Results are presented in Table 7.

| Modification ratio | 5 % | 10 % |
|---|---|---|
| Average accuracy rate | 86.56 % | 78.36 % |
| Average weighted accuracy rate | 95.19 % | 86.18 % |
| Average run time | 27 mn 8 s | 77 mn 17 s |

Table 7: Results of the synthetic data alignment with the second noise model

It can be observed that average accuracy rates decrease significantly but weighted accuracy rates keep better scores. It must be kept in mind that difference rates between texts are 20 and 40 % instead of 15 and 30 % with the first noise model. Further, the difference between weighted and non-weighted accuracy rates indicates that invariant block have a better classification rate than other blocks. This is confirmed in Table 8 which presents the average confusion matrices: reference alignment labels are in lines and MEDITE alignment labels in columns. Main errors come from the identification of moves as insertions or deletions; but a move can be considered as a deletion plus an insertion. Insertions and deletions are also mistaken for replacements, but replacements can also be considered as a deletion plus an insertion. This implies that it is a decision problem: our decision model is very simple for replacements (see Section 3) and may be improved.

## 5 Conclusion and future work

This paper presents MEDITE, the application we developed to address a pragmatic problem issued from textual genetic criticism. Our sequence alignment algorithm detects moves and enables to identify and link duplicate character blocks between two texts. The algorithm is compared with similar ones in four experiments. It surpasses them for very noisy text alignment, a very difficult task. It was not designed for duplicate linkage and text reuse detection, nevertheless it is general enough to handle these tasks without the need of any adaptation. It can be very useful in more specialized applications of these last two tasks. Our results are less good than those of [Feng and Manmatha, 2006] but our algorithm is more general because it handles moves and could be adapted easily.

We are working with a team dealing with machine translation on an ongoing experiment similar to the task of paraphrase extraction presented in [Regina Barzilay, 2001] (see Section 2). The goal is to align a machine translation output with a reference text in order to identify machine translators' errors and to correct them automatically. We also plan to adapt the algorithm for medieval text alignment where word spelling is not well-established and can change between two text versions according to the copyists' mood.

## Acknowledgments

We would like to thank the anonymous referees for their many useful comments.

| Modification ratio | 5 % | | | | | 10 % | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Inv. | Ins. | Del. | Repl. | Mov. | Inv. | Ins. | Del. | Repl. | Mov. |
| Invariants | 98.07 | 0.56 | 0.52 | 0.76 | 0.08 | 94.0 | 1.75 | 1.6 | 2.32 | 0.32 |
| Insertions | 0.21 | 92.16 | 0 | 7.55 | 0.07 | 0.21 | 85.1 | 0 | 14.52 | 0.17 |
| Deletions | 0.14 | 0 | 87.76 | 9.25 | 1.76 | 1.46 | 0 | 76.77 | 17.78 | 3.99 |
| Replacements | 0.70 | 5.40 | 4.65 | 88.40 | 0.84 | 0.72 | 11.32 | 9.43 | 76.34 | 2.18 |
| Moves | 1.43 | 13.70 | 14.05 | 4.38 | 66.43 | 1.47 | 15.16 | 15.55 | 8.25 | 59.56 |

Table 8: Average confusion matrices (in %) for the synthetic data alignment with the second noise model

## References

[Arslan *et al.*, 2001] Abdullah N. Arslan, Omer Egecioglu, and Pavel A. Pevzner. A new approach to sequence comparison: normalized sequence alignment. *Bioinformatics*, 17(4):327–337, 2001.

[Bergroth *et al.*, 2000] L. Bergroth, H. Hakonen, and T. Raita. A Survey of Longest Common Subsequence Algorithms. In *SPIRE '00: Proceedings of the Seventh International Symposium on String Processing Information Retrieval*, 2000.

[Bourdaillet and Ganascia, 2006] Julien Bourdaillet and Jean-Gabriel Ganascia. MEDITE: A Unilingual Textual Aligner. In Tapio Salakoski et al., editor, *Advances in Natural Language Processing, 5th International Conference on NLP, FinTAL 2006, Turku, Finland, Proceedings*, volume 4139 of *Lecture Notes in Artificial Intelligence*, pages 458–469. Springer, 2006.

[Bray *et al.*, 2003] Nick Bray, Inna Dubchak, and Lior Pachter. AVID: A Global Alignment Program. *Genome Res.*, 13(1):97–102, 2003.

[Chiao *et al.*, 2006] Yun-Chuang Chiao, Olivier Kraif, Dominique Laurent, Thi Minh Huyen Nguyen, Nasredine Semmar, François Stuck, Jean Véronis, and Wajdi Zaghouani. Evaluation of multilingual text alignment systems: the ARCADE II project. *Proceedings of the LREC 2006 Conference*, 2006.

[Clough *et al.*, 2002] Paul Clough, Robert Gaizauskas, Scott Piao, and Yorick Wilks. METER: MEasuring TExt Reuse. *In proceedings of the 40th Anniversary Meeting for the Association for Computational Linguistics (ACL-02)*, pages 152–159, 2002.

[Deppman *et al.*, 2004] Jed Deppman, Daniel Ferrer, and Michael Groden, editors. *Genetic Criticism - Texts and Avant-textes*. University of Pennsylvania Press, 2004.

[Feng and Manmatha, 2006] Shaolei Feng and R. Manmatha. A hierarchical, HMM-based automatic evaluation of OCR accuracy for a digital library of books. In *JCDL '06: Proceedings of the 6th ACM/IEEE-CS joint conference on Digital libraries*, pages 109–118, New York, NY, USA, 2006. ACM Press.

[Gusfield, 1997] Dan Gusfield. *Algorithms on Strings, Trees and Sequences: Computer Science and Computer Biology*. Cambridge University Press, 1997.

[Kolman *et al.*, 2004] Petr Kolman, Avraham Goldstein, and Jie Zheng. Minimum Common String Partition Problem: Hardness and Approximations. In *Proceedings of the 15h International Symposium on Algorithms and Computation (ISAAC)*, 2004.

[Levenshtein, 1966] V.I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversal. *Cybernetics and Control Theory*, 10(8):707–710, 1966.

[Lopresti and Tomkins, 1997] Daniel P. Lopresti and Andrew Tomkins. Block Edit Models for Approximate String Matching. *Theoretical Computer Science*, 181(1):159–179, 1997.

[Myers, 1986] Eugene W. Myers. An O(ND) Difference Algorithm and Its Variations. *Algorithmica*, 1(2):251–266, 1986.

[Needleman and Wunsch, 1970] Saul Needleman and Christian Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443–453, 1970.

[Regina Barzilay, 2001] Kathleen McKeown Regina Barzilay. Extracting Paraphrases from a Parallel Corpus. *in Proceedings of ACL/EACL*, 2001.

[Shapira and Storer, 2002] Dana Shapira and James A. Storer. Edit Distance with Move Operations. In Alberto Apostolico and Masayuki Takeda, editors, *CPM*, volume 2373 of *Lecture Notes in Computer Science*, pages 85–98. Springer, 2002.

[Smit *et al.*, 1996] A.F.A. Smit, R. Hubley, and P. Green. RepeatMasker Open-3.0, 1996.

[Smith and Waterman, 1981] Temple F. Smith and Michael S. Waterman. Identification of Common Molecular Subsequences. *Journal of Molecular Biology*, 147:195–197, 1981.

[Tichy, 1984] Walter F. Tichy. The String-to-String Correction Problem with Block Moves. *ACM Trans. Comput. Syst.*, 2(4):309–321, 1984.

[Ukkonen, 1995] Esko Ukkonen. On-Line Construction of Suffix Trees. *Algorithmica*, 14(3):249–260, 1995.

[Wilbur and Lipman, 1984] W. J. Wilbur and David J. Lipman. The context dependent comparison of biological sequences. *SIAM J. Applied Mathematics*, 44(3):557–567, 1984.

[Wise, 1996] Michael J. Wise. YAP3: Improved Detection of Similarities in Computer Program and Other Texts. *SIGCSE Bulletin (ACM Special Interest Group on Computer Science Education)*, 28, 1996.