Comparing Sanskrit Texts for Critical Editions: 
the sequences move problem. *

Nicolas Béchet¹, Marc Le Pouliquen², and Marc Csernel³,⁴

¹ GREYC Université de Caen Basse-Normandie, Campus II science 3
14032 Caen CEDEX, France.
nicolas.bechet@unicaen.fr

² Telecom Bretagne, Labsticc UMR 3192, BP 832, 29285 Brest Cedex - France
Marc.LePouliquen@univ-brest.fr

³ INRIA, Projet AXIS, Domaine de Voluceau, Rocquencourt BP 105
78153 Le Chesnay Cedex, France
Marc.Csernel@inria.fr

⁴ Université Paris Dauphine, 1 Place du Maréchal DeLattre de Tassigny
75016 Paris, France.

Abstract. A critical edition takes into account various versions of the same text in order to show the differences between two distinct versions, in terms of words that have been missing, changed, omitted or displaced.

Traditionally, Sanskrit is written without spaces between words, and the word order can be changed without altering the meaning of a sentence. This paper describes the characteristics which make Sanskrit text comparisons a specific matter. It presents two different methods for comparing Sanskrit texts, which can be used to develop a computer assisted critical edition. The first one method uses the L.C.S., while the second one uses the global alignment algorithm. Comparing them, we see that the second method provides better results, but that neither of these methods can detect when a word or a sentence fragment has been moved.

We then present a method based on N-gram that can detect such a movement when it is not too far from its original location. We will see how the method behaves on several examples and look for future possible developments.

1 Introduction

A critical edition is one that takes into account all the different known versions of the same text. If the text is mainly known through a great number of manuscripts that include non trivial differences, the critical edition often looks rather daunting for readers unfamiliar with the subject: the edition is mainly made up by footnotes that highlight the differences between manuscripts, while the main text (that of the edition) is rather short, sometimes just a few lines per page. The differences between the texts are usually described in terms of words (sometimes sentences) that are missing, or have been added or changed in a specific manuscript. This bring up to mind the edit distance but

* This work have been supported by the EEC FP7 programm IDEAS
in term of words rather than characters. The text of the edition is established by the editor according to his own knowledge of the text. It can be a particular manuscript or a ‘mean’ text built according to some specific criteria. Building a critical edition by comparing texts one with another, especially manuscript ones, is a task which is certainly long and, sometimes, tedious. This is why, computer programs have long been helping philologists in their work (see [11] or [10] for example), but most of them are dedicated to texts written in Latin (sometimes Greek) scripts.

In this paper we focus on the problems involved by a critical edition of manuscripts written in Sanskrit. Our approach is illustrated by texts that are extracted from manuscripts of the “Banaras gloss”: the kāśikāvṛtti.

The Banaras gloss was written around the 7th century A.D., and is one of the most famous commentaries on Pāṇini’s grammar, which is known as the first generative grammar ever written, and was written around the fifth century B.C. as a set of rules. These rules cannot be understood without the explanations provided by a commentary such as the kāśikāvṛtti. This corpus was chosen, because it is one of the largest collection of manuscripts (about hundred different ones) of the same text actually known. Notice that, since some manuscripts have been damaged by mildew, insects, rodents, etc., they are not all complete.

In what follows we first describe the characteristics of Sanskrit that matter for text comparison algorithms, we will then show that such a comparison requires the use of a lemmatized text as the main text. Using a lemmatized text induces the need of a lexical preprocessing. Once the lexical preprocessing has been carried out, we can proceed to the comparison, where two approaches have been developed, the first one based on the Longest Common Subsequence (L.C.S.) by [4], and the second one on edit distance by [3]. The second method is easier to use, so we use it to align our Sanskrit texts before moving sequences.

Because the word order is not always meaningful in Sanskrit, some manuscripts have some words sequences which are not in the same place than in the main text, and the alignment procedure, whichever it is, is not able to align such sequences together. The misplaced sequence appears to be missing in one place, and added in another one. This is why we use here a word n-gramm based method, to discover if some sequence moves are likely to exist, and then determine their precise limits, where they have been moved, and display them. This improvement is detailed in the section 4 of this paper. Remark that the sequence move problem is quite similar to the translocation problem which exists in genomics.

2 How to compare Sanskrit manuscripts

2.1 Sanskrit and its graphical characteristics

One of the main characteristics of Sanskrit is that it is not linked to a specific script. Here however we provide all our examples using the Devanāgarī script, which is nowadays the most most widely used. The script has a 48-letter alphabet.

Due to the long English presence in India, a tradition of writing Sanskrit with the Latin alphabet (a transliteration) has long been established and used by many European

scholars such as Franz Bopp in 1816. All these transliteration schemes were originally carried out to be used with traditional printing. It was adapted for computers by Frans Velthuis [13], more specifically to be used with \TeX. According to the Velthuis transliteration scheme, each Sanskrit letter is written using one, two or three Latin characters; notice that all our corpus is written according to the Velthuis scheme and not in Devanāgarī Unicode [2].

In ancient manuscripts, Sanskrit is written without spaces, and this is an important graphical specificity, because it greatly increases the complexity of text comparison algorithms.

On the other hand, each critical edition deals with the notion of word. Since electronic Sanskrit lexicons such as the one built by Huet [7, 6] do not cope with grammatical texts, one must find a way to identify each Sanskrit word within a character string, without the help of either a lexicon or of spaces to separate the words.

The reader interested in exploring deeper approach of the Sanskrit characteristics which matter for a computer comparison can refer to [4].

2.2 How to proceed?

The solution comes from the lemmatization of one of the two texts of the comparison: the text of the edition. The lemmatized text is prepared by hand by the editor. It is called padapāṭha, according to a mode of recitation where syllables are separated. From this lemmatized (the padapāṭha) text, we will build the text of the edition, which is called saṃhitapāṭha, according to a mode of recitation where the text is said continuously. The transformation of the padapāṭha into the saṃhitapāṭha is not straightforward because of the existence of sandhi rules.

What is called sandhi — from the Sanskrit: liaison — is a set of phonetic rules which apply to the morpheme junctions inside a word or to the junction of words in a sentence. These rules are perfectly codified in Pāṇini’s grammar. Roughly speaking, written Sanskrit reflects (via the sandhi) the liaison(s) which are made by a human speaker. A text with separators (such as spaces) between words, can look rather different (the letter string can change greatly) from a text where no separator is found.

An example of padapāṭha:

\[ \text{vi"ud"panna_ruupa_siddhi+v.rtti+iya.m kaa"sikaa_naama} \]

We can see that words are separated by spaces and three different lemmatization signs: +, -, ^.

The previous padapāṭha becomes the following saṃhitapāṭha:

\[ \text{vyutpannaruupasiddhivr.rtti+iya.mkaa"sikaanaama} \]

the bold letters represent the letters (and the lemmatization signs) which have been transformed, according to a sandhi rule.

We call the typed text, corresponding to each manuscript: Typed Manuscript (T.M.). Each T.M. contains the text of a manuscript and some annotation commands. The annotation commands keep trace of all the modifications of the manuscript not explicitly present in the text, such as change of ink color, a hole made by a rodent, etc.. They provide a kind of meta-information. Each manuscript is typed by a scholar.
The processing is done in four steps, but only two of them will be considered in this paper:

- **First step**: A lexical preprocessing. The *padapāth* is transformed into a virtual *samhitapāth* in order to make a comparison with a *T.M.* feasible. The transformation consists in removing all the separations between words and then in applying the *sandhi*. This virtual *samhitapāth* form the text of the edition.

- **Second step**: An alignment of a *T.M.* and the virtual *samhitapāth*. The aim is to identify, as precisely as possible, the words in the *T.M.*, using the *padapāth* as a pattern.

- **Third step**: Once the alignment has been achieved and the words of the *T.M.* have been determined, try to improve the alignments results. Determine which word have been added suppressed changed, or moved.

- **Fourth step**: Display the results in a comprehensive way for the editor. This step is accomplished using XML.

The comparison is done paragraph by paragraph.

### 2.3 Why not use the diff algorithm

The very first idea to compare Sanskrit text was to use *diff* in order to obtain the differences between two Sanskrit sequences.

But the results related in [4] were quite disappointing. The classical *diff* command line provided no useful information at all. The result of the comparison of the two following sequences: "srii ga.ne"saaya nama.h and *tasmai* "srii gurave namas* just said that they were different.

This is why [4] started to implement their own L.C.S. based algorithm. Its results appear in the right-hand column of Table 1. We can see that they are expressed in term of words.

<table>
<thead>
<tr>
<th>lcl</th>
<th>1d0</th>
<th>Word 1 ‘tasmai’ is:</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;sriigane&quot;saayanama.h</td>
<td>tasmai</td>
<td>- Missing</td>
</tr>
<tr>
<td>&gt; tasmai&quot;sriiguravena.h</td>
<td>gurave</td>
<td>- Followed by Added word(s) ‘ga.ne&quot;saaya‘</td>
</tr>
<tr>
<td>&gt; gane</td>
<td>&quot;saaya</td>
<td>- Missing</td>
</tr>
<tr>
<td>diff without space</td>
<td>ediff with space</td>
<td>L.C.S. based results without space</td>
</tr>
</tbody>
</table>

**Table 1**: Example of results with L.C.S.

### 2.4 The L.C.S based method

This method was developed by [4], and was the first method used to build critical edition of Sanskrit texts. The L.C.S matrix associated with the previous result can be seen in figure 1(p. 5). In this figure the vertical text represents the *samhitapāth*, the horizontal

The horizontal bold dark lines were provided by the padapātha, before it was transformed into the samhitapātha.

The rectangles indicate how the correspondences between the samhitapātha and the T.M. were done. One corresponds to a missing word (tasmai) two correspond to a word present in both strings the words (śrī and nāmaḥ), and the last one corresponds to a word with a more ambiguous status, we can say either that the word has been replaced or that one word is missing and another word has been added.

Fig. 1: The L.C.S. Matrix

If the result appears quite obvious within this example, it is not always so easy, particularly when different paths within the matrix can lead to different alignments providing different results. This led the authors to look for some navigation rules which are precisely related in [4].

3 The alignment based on edit distances

This method has been developed by [3] because the L.C.S based method needed too much post processing to be really efficient and because the lack of “substitution” relative to L.C.S method was really a handicap when just one or two letters were changed in a word. The method is still not perfect but is a better base than L.C.S. to provide in the simplest way good alignments, but some improvements still need to be done before this method can be used effectively. Building an alignment using the edit distance matrix, starts by the low right-hand corner, going to the upper left. So the letters of the of the T.M. are aligned on the first letter of the samhitapātha with the same value they meet coming from the end of the paragraph. Sometimes this procedure induces the presence of Orphan letters which will be moved to obtain a better alignment as explained in the next subsection.
3.1 Shifting the orphan letters

We call an orphan letter an isolated letter belonging to an incomplete word of (usually) a manuscript. To obtain a proper alignment, these letters must fit with the words to which they belong.

Table 2: An example of orphan letters

Table 3: Orphan letters after being shifted

Table 2 gives a good example. The upper line of the table represents the padapātha, the lowest one a T.M. The word separation induced by the padapātha are indicated by double vertical lines. Because the padapātha is used as a template the separations appears also within the T.M.. The orphan letters appears in bold. The words pratyāhaaraa and rtha.h are missing in the T.M.. Consequently the letters a.h are misplaced, with the word rtha.h. The goal is to shift them to the right place with the word upade"sa.h. The result after shifting the letters appears in Table 3. The bold letters are the letters which has been shifted.

In the second example (Table 4) we see on the left side of the table that the letter a must just be shifted from the beginning of asyddhy to the end of saavarny giving the right-hand part of the table.

Table 4: Another example

Table 5: Shifting within the padapātha
3.2 Measuring the quality of the alignment.

It is difficult to find a unique method to measure the quality of the alignments, because each method (L.C.S. and Edit Distance) produces a different type of alignment where the improvement must be made in a different way. L.C.S. methods do not induce any substitution, so measuring of the quality is quite easy to find: more "empty" characters we have, the lower the quality is; even if it can be nuanced in some particular cases. On the other hand with the edit distance based method, the algorithm provides substitutions and we can observe that the more substitutions we have, the lower quality. Remark that all substitutions are not irrelevant, and should not lower the quality, but some of them do, mainly the one corresponding to a possible translocation. Missing or added letters obtained by the edit distance method have always been relevant. Because an important part of the substitution are relevant we will never obtain a null score. We know we have to improve our measure.

4 An improvement of the alignment by an n-gram based method.

The goal of this approach is to improve the alignment (obtained by edit distance method) of the samhitapātha and a T.M. once the orphan letters have been shifted. An example of an alignment which can be improved is written in the table 6. The bold letters are letters which have been substituted, and lowered the quality.

We define the alignment score by the number of substituted letters, better the alignment, lower the score. We obtain with this alignment a score of 13, which is poor. This score can be improved if we refer to the complete sentence in the samhitapātha and the T.M.: we can then move the sequence la kaare tv of the T.M. by inserting empty letters on the left (i.e. the '-' symbol), in the T.M. and on the right in the samhitapātha we obtain a good alignment (with a score of 0\(^5\)) by aligning la kaare tv in a manuscript with la kaare tv of the samhitapātha as in the alignment below in table 7. The bold letters indicate the sequence moved.

\(^5\) Note that an insertion (noted '-' ) is not lowering the score with the edit distance alignment.
4.1 Overview of the procedure

It is described by the steps below, which are detailed in the next subsections.

1. We first extract word n-gram from the *samhitapātha* (cf § 4.2).
2. We search for each word n-gram all the possible better alignments (called a *candidate*) in the T.M.
3. For each candidate, we modify the original alignment to take the candidate into account, and build new word borders according to the *samhitapātha* (cf § 4.3).
4. We apply optimizations to the new alignment (cf § 4.4).
5. For each new generated alignment, we build a new score and write a proposition of improvement.

4.2 Extracting the word n-gram

The first step is to extract *word n-gram*. Word n-gram are frequently employed in the literature, one of the first uses being [12], or more recently in [9]. An n-gram of $X$ can be defined as a sequence of $n$ successive $X$ with $X \in \{\text{word, letter}\}$. The figure 2 provides an example of letter 2-gram and word 3-gram extraction.

![Fig. 2: Example of n-gram extraction.](image)

We use word n-gram extraction in the hope to place all occurrences of a word in a unique context. A single word can have many occurrences in the *samhitapātha*, but only one when it is preceded and followed by some specific words. The word *kaare* in table 8 has a lot of instances but only one which is preceded by word *la* and followed by the word *tv*. Thus, by taking a 3-gram, we obtain only one instance of the word 3-gram *la kaare tv* and our extraction becomes relevant.

We use the extraction to make a (word) segmentation of the *samhitapātha*. An example of 4-gram from the *samhitapātha* is given in the upper line of Table 8, the lower line contains letters which do not fit the 4-gram, and that we will try to move, the bold letters are well aligned. In the following when we talk about *word n-gram*, we always refer to the *samhitapātha*.

| nub-an dh-a | la-kaare-te tv |
| re-v ak aar a h | it-s a m j n o |

Table 8: 4-gram alignment example with a score 13.
The $n$ value is set by the user, but could, in the future, be computed by program.

### 4.3 The search for candidates

*Candidates* are defined as strings in a manuscript which could provide a better alignment for a given word $n$-gram. Thus, candidates must contain as many letters than the word $n$-gram. In figure 3, the word $n$-gram `nub-andh-a.h la kaare tv` has 16 letters (without spaces). Then, all candidates must have 16 letters. To get them we shift a sliding window of 16 letters, from left to right, and right to left from the initial position. The number of shifting is given by the parameter $m$; in the example shown in figure 3, the parameter $m = 4$: we have shifted the sliding window of 4 positions to the left.

To estimate the new alignment quality we compute the score, as defined in section 4: the number of substitutions made between the two strings (the one from the *samhitapātha* is the current word $n$-gram, the other one from the *T.M.* is a candidates. The alignment of strings: `nub-andh-a.h la kaare tv` and `retvakaara.h it sa.mj `no` in Table 8 obtain, for instance, a score of 13.

If a candidate provides a better score that the original one, we keep it. At the end, we keep the best, the one with the lowest score. If several candidates have the same score, we keep the last one, but this situation never occurred with the examples we have considered.

---

6 An $m$ value of 4 means that 8 candidates are tested, 4 from left $m$-letter $l$-gram and 4 from right $m$-letter $l$-gram.
To summarize, we first need to compute the score of the original alignment (13). Then, the original word n-gram nub-andh-a.h la kaare tv is compared with the 4 candidates:

- aaretvakaara.hit sa.mj^n,
- kaaretvakaara.hitsa.mj,
- akaaretvakaara.hitsa.m,
- lakaaretvakaara.hitsa

The word separator ‘ ’ is not reported because word separation is no longer relevant (i.e. words are no longer aligned). At the end, with these four candidates, none of the scores is better than 13, the four candidates must be rejected.

4.4 Integrating the best candidates

After the selection of the best candidate for a given word n-gram, we need to build new word boundaries. Let us consider the example in Table 8 with the word n-gram nub-andh-a.h la kaare tv. With the parameter $m = 20$, we find caara.naartha.hlakaaretv as the best candidate. To align this candidate with the word n-gram we need first to adjust the corresponding word boundaries.

**Aligning the n-gram and the candidate.** In the following tables, the n-gram and it corresponding best candidate are in bold:

<table>
<thead>
<tr>
<th>nub-andh-a.h la kaare tv</th>
<th>caara.naartha.hlakaaretv</th>
</tr>
</thead>
</table>

Table 9: the word 4-gram in the *samhitapātha*

<table>
<thead>
<tr>
<th>nub-andh-a.h la kaare tv</th>
<th>caara.naartha.hlakaaretv</th>
</tr>
</thead>
</table>

Table 10: the best candidate in a manuscript

We need to adjust the two texts to align nub-andh-a.h la kaare tv with caara.naartha.hlakaaretv. This adjustment is made by adding insertions of empty characters both in the *samhitapātha* and the T.M.

The number of insertions is equal to the absolute value of the substraction of the position of the first letter of the word n-gram and the position of the first letter of the candidate. In our example, the first letter of the word n-gram is 24 and the position of the first letter of the candidate is 10. Thus, we need to insert 14 letters. We obtain the alignment described in Table 11, in this table the dots represent the places where the insertions have been done. The number of dot is not relevant. In the sequence corresponding to a manuscript, the word limits have disappeared, because they are no more relevant, we will reconstruct them on the next step.

<table>
<thead>
<tr>
<th>nub-andh-a.h la kaare tv</th>
<th>caara.naartha.hlakaaretv</th>
</tr>
</thead>
</table>

Table 11: The alignment before the word limits reconstruction
The insertion made is relative to the position of the word n-gram compared to the position of the candidate. If the candidate precedes the word n-gram (as in our example), we first need to shift the T.M. and then the *saṃhitapātha*, otherwise we shift the *saṃhitapātha* first.

**The *saṃhitapātha* words alignment.** After the candidates has been correctly aligned with the word n-gram, we need to align the existing words of the word n-gram with the unsegmented candidate, according to the *saṃhitapātha* words. For example, with the example from Table 8, the new word segment appears in the Table 12. The alignment represented is not a good one, he still has a score = 6, which can be ameliorated by the following step, the bold letter are the letters which has been substituted.

<table>
<thead>
<tr>
<th>n</th>
<th>a</th>
<th>n</th>
<th>dh</th>
<th>r</th>
<th>a</th>
<th>s</th>
<th>a</th>
<th>h</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>s</td>
<td>n</td>
<td>a</td>
<td>r</td>
<td>a</td>
<td>s</td>
<td>a</td>
<td>h</td>
</tr>
<tr>
<td>c</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>n</td>
<td>a</td>
<td>a</td>
<td>r</td>
</tr>
</tbody>
</table>

Table 12: The newly reconstructed word boundaries

We can now consider the new word boundaries and report them within the T.M.

### 4.5 Optimizing the new alignments

Making a new alignment on texts can induce new orphan letters. Thus, we apply the approach presented in Section 3.1 to obtain a better alignment. For instance, with our previous example, we identify many orphan letters as 'c', 'aa', 'r', etc. on the T.M. as shown below:

<table>
<thead>
<tr>
<th>n</th>
<th>a</th>
<th>n</th>
<th>dh</th>
<th>r</th>
<th>a</th>
<th>s</th>
<th>a</th>
<th>h</th>
</tr>
</thead>
<tbody>
<tr>
<td>c</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>n</td>
<td>a</td>
<td>a</td>
<td>r</td>
</tr>
<tr>
<td>c</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>n</td>
<td>a</td>
<td>a</td>
<td>r</td>
</tr>
</tbody>
</table>

Table 13: Identification of the new orphan letters.

<table>
<thead>
<tr>
<th>n</th>
<th>a</th>
<th>n</th>
<th>dh</th>
<th>r</th>
<th>a</th>
<th>s</th>
<th>a</th>
<th>h</th>
</tr>
</thead>
<tbody>
<tr>
<td>c</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>n</td>
<td>a</td>
<td>a</td>
<td>r</td>
</tr>
<tr>
<td>c</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>n</td>
<td>a</td>
<td>a</td>
<td>r</td>
</tr>
</tbody>
</table>

Table 14: The new orphan letters after a shift.

### 4.6 Building the final score

The final step of our approach is the final score of the new alignment computation, according to the whole environment (the neighboring words) and it delivery to an expert. The new computed score is quite different from the previous one used to select the candidates. The definition is the same: the number of substitutions made between the *saṃhitapātha* and the T.M. The difference lies in the texts we use for the comparison. For candidate selection, we only focus on a word n-gram and a candidate, which has the same number of letters. For the final score we use the new alignment.

### 4.7 Experiments

**Experimental protocol.** We use as a corpus the first chapter of the *kāśikāvṛtti*: the *Pratyāhāra sutra*, and we have used only 8 manuscripts (T.M.) randomly chosen from...
the fifty available. Each manuscript contains a maximum of 34 paragraphs. We first apply the alignment method, then after shifting the orphan letters, we obtain a baseline alignment. Our goal is to improve the baseline by lowering the number of letter substitutions. Recall that with edit distance, the more letter substitutions we have, poorer the alignment, higher the score.

Then we apply our approach. Considering \( P \) paragraphs, \( K \) manuscript, \( w \) maximum words by paragraphs, which means \( w - (n-1) \) word \( n \)-gram by paragraphs, and \( m \) tests of possible candidates for each word \( n \)-gram, the maximum numbers of operation \( Nb \) to apply our method is approximated by:
\[
Nb = P \times K \times m \times (w - (n - 1)).
\]
We have a complexity depending mostly on the \( n \) of \( n \)-gram, the number of paragraphs, the number of manuscripts, and the number of possible candidates to be tested.

For instance, with the parameters \( P=34, K=8, n=4, m=40 \), we obtain:
\[
Nb = 34 \times 8 \times 40 \times (w - (4 - 3)) = 10,880 \times (w - 3).
\]

<table>
<thead>
<tr>
<th>( n ) (words)</th>
<th>( m )</th>
<th>( Nb )</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseline</td>
<td></td>
<td>1,110</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>40</td>
<td>10,880</td>
<td>922</td>
</tr>
<tr>
<td>2</td>
<td>40</td>
<td>10,880</td>
<td>908</td>
</tr>
<tr>
<td>3</td>
<td>40</td>
<td>10,880</td>
<td>896</td>
</tr>
<tr>
<td>4</td>
<td>40</td>
<td>10,880</td>
<td>843</td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td>10,880</td>
<td>1,080</td>
</tr>
<tr>
<td>3</td>
<td>60</td>
<td>10,880</td>
<td>784</td>
</tr>
<tr>
<td>3</td>
<td>80</td>
<td>10,880</td>
<td>784</td>
</tr>
</tbody>
</table>

Table 15: Experimental results.

Experimental results.
The results are displayed in Table 15. The baseline obtains a score of 1,110. It means that 1,110 substitutions were made during the construction of the alignment of sanhita-pāṭha and the 8 different manuscripts.

We first want to see the influence of the \( n \) parameter. According to our experiments, the best score is obtained with \( n = 3 \) allowing 212 substitutions to be suppressed from the baseline. However, the \( n \) parameter does not seem to have an important influence on the scores.

In comparison, a modification of \( m \) parameter provides a better score with only 784 substitutions. It seems that 326 substitutions were deleted compared with the baseline. Note that we obtain the same score for \( m = 60 \) and \( m = 80 \); \( m = 60 \) is better because we need fewer operations to obtain the same quality of alignment.

5 Conclusion and future work

5.1 Future Improvement

Our first trials made us discover some unexpected situations that we do not take into account. Consider the alignment produced by our program in table 16, where the bold
letters are orphan letters unusually placed in the *samhitapātha*. We need to shift them to get a better result which can be seen in table 17, where the italic letters refer to letters concerned by our second improvement.

![Table 16: An unexpected alignment.](image)

Because of the bad alignment induced by the orphan letters in the *samhitapātha* our n-gram based method was unable to discover that the sequences in italic letters of figure 17 was corresponding to a translocation, and then did not made the move which lead us to the right alignment such as displayed in the table 18, where the bold letters correspond to the letters newly aligned.

![Table 17: The alignment after shifting the orphan letters of the *samhitapātha*.](image)

To solve this problem, it seems at first sight that we need only to take a better care of the orphan letters in the *samhitapātha*, but we need a real trial to check if this simple action will be sufficient.

We can remark that the use of n-gram is not the only possible way to solve the translocation (sequence move) problem. A problem quite similar exists in genomics, and can be solved by methods such as "Glocal alignment" [1]. These methods do not follow the same goal than we do, but they may be source of inspiration for our future work.

![Table 18: New words alignment obtained by a sequence move.](image)

### 5.2 Improvement of the score computation

The score we use provides good results according to our purpose. But we would like to get a criterion which can provide a value near from zero when the two texts compared differ only by the usual differences that one can expect when comparing two different versions of the same text. Two reach this goal we will have to go step by step to drop out some elements that we don’t want to take into account for the score.
For example some Sanskrit letters are prone to be confused by a scribe, so they need a special treatment, together for our score computation and in our alignment method. A second step could be a special treatment for the substitution of sparse letters which is obviously due to a different cause than a long sequence substitution.

On the other hand we have to build a second critter which can be used with a L.C.S. If we succeed on both of these goal we will be able to compare the two approaches (L.C.S. and edit distance) with a measure.

5.3 Using the critical edition as a distance

While the definition of a critical edition reminds us the one of the edit distance, we can build a distance between two manuscripts using the sanhitapāṭha as an intermediary. The easiest is to build a binary sequence representing each manuscript, where each word of the sanhitapāṭha is represented by a one, when it is present in the manuscript, and by a zero when it is absent. Using such a manuscript representation, we can build a distance between two manuscripts using a method such as the Hamming distance[5]. From this distance we can build a phylogenetic tree or Stemma codicum [8] between the different manuscripts to discover if some of them are deriving from the others. But one must remark that the translocation can not be taken directly into account by this method. We have to conduct some experiment to check if the sequence move must be considered as relevant in the distance construction, and if a distance construction which takes it into account does not induce a significant amount of complication.

5.4 Conclusion

By solving the sequence move (translocation ) problem we have improved in a significant way an automated construction of Critical edition of Sanskrit text, opening the way to a more efficient construction of a phylogenetic tree.

One can remark that even after detection of the translocations, our score, which measures the quality of our alignment, is far to be null, which it should be if the quality is at his best. This is due to the nature of a critical edition, there are natural changes between our different texts and some of these changes are reflected in a natural way by letter substitutions which increase the score. Even if our score computation provides us valuable information concerning the sequence move, it surely must be improved, and completed by a score adapted to a L.C.S. alignment.

The differences enlightened by construction of critical edition are a very good base for the construction of a distance between the different manuscripts, and the provide a solid base to build a Stemma codicum. The construction of a Stemma codicum is always a kind of grail for a philologist, we have the hope that the amelioration we have done in the comparison of the Sanskrit texts will lead to a better Stemma.

References


