# Comparaison de manuscrits sanskrits

## (édition critique et classification)

**Marc Csernel[1], Patrice Bertrand[2]**

*[1]Inria Rocquencourt & Université Paris IX Dauphine*
*Domaine de Voluceau ; BP 105*
*78 153 Le Chesnay Cedex ; France*
*Marc.Csernel@inria.fr*

*[2]GET - ENST Bretagne ; Umr Tamcic & Dept Lussi*
*Technopôle Brest-Iroise ; CS 83818*
*29 238 Brest Cedex 3 ; France*
*Patrice.Bertrand@enst-bretagne.fr*

**Résumé :**

*Face à une collection de manuscrits retranscrivant un même texte, fréquemment dispersés dans le temps et l'espace, le chercheur a recours à l'édition critique afin de prendre en compte tous les aspects du texte. En vue de faciliter leurs créations et d'offrir de nouvelles possibilités en matière d'analyse et d'interactivité, quelques éditions critiques informatisées ont déjà été proposées. Dans cet article, nous nous intéressons au cas de l'édition critique (électronique) de manuscrits écrits en sanskrit. Nous commençons par décrire les spécificités du sanskrit qui soulèvent un ensemble de problèmes touchant aussi bien à la typographie qu'à l'informatique et l'analyse de données. Puis, nous présentons brièvement les solutions informatiques actuelles permettant d'adapter les traitements à la typographie et la syntaxe particulière au sanskrit. Enfin, nous proposons une approche automatique pour identifier et évaluer les différences entre deux manuscrits en vue notamment de décrire les relations de filiation entre manuscrits connus d'un même texte. D'un point de vue algorithmique, notre approche est basée sur des techniques habituellement employées pour comparer des chaînes moléculaires.*

**Mots clés:**

*édition critique, sanskrit, distance intertextuelle, arbres phylogénétiques.*

## 1. Introduction

Une édition critique est un ouvrage qui fait apparaître toutes les différences existantes entre les variantes d'un texte. L'élaboration d'une édition critique prend toute son importance et revêt toute sa difficulté lorsqu'un texte est connu au travers d'un vaste ensemble de manuscrits dispersés à la fois dans le temps et dans l'espace. En effet**,** la retranscription des manuscrits est affectée non seulement par des modifications successives provenant de façon volontaire ou non des scribes, mais aussi par des atteintes du temps qui peuvent rendre inutilisable telle ou telle partie. En outre, il est possible qu'un manuscrit provienne de plusieurs sources : manuscrits recopiés par d'autres scribes, épigraphies, ...

Si l'édition critique comprend de nombreux manuscrits, elle peut alors revêtir une allure rébarbative pour le profane, puisqu'en poussant les choses à l'extrême, le lecteur se retrouve devant une page comportant quelques lignes faisant partie du texte de l'édition (appelé encore texte "maître"), et un grand nombre de lignes écrites sous la forme de notes de bas de pages, décrivant par le menu les variations du texte en fonction des différentes sources possibles.

Le choix du texte d'une édition critique relève de l'éditeur qui, après comparaison des manuscrits, peut aussi bien retenir un manuscrit particulier que proposer un texte moyen, obtenu à partir de plusieurs manuscrits. La comparaison deux à deux des variantes d'un texte étant un travail des plus fastidieux, le choix du texte de l'édition est donc un authentique travail de bénédictin.

C'est pourquoi depuis longtemps l'informatique s'est mise au service des "éditeurs critiques". Malheureusement l'écriture de la langue sanskrite est dotée de caractéristiques qui rendent inutilisables les logiciels d'édition habituels. Par exemple et bien qu'il puisse être appliqué au sanskrit, Edmac [LAV 96] n'est qu'un ensemble de macros Tex qui facilitent la présentation de l'édition. Malgré ses succès concernant l'anglais [OHA 93], Collate [ROB 94,00] n'a pas réussi à donner des résultats satisfaisants en ce qui concerne le sanskrit. Le logiciel Anastasia [ANA 00] donne à l'université de Munster de bons résultats avec la graphie grecque en fournissant une édition critique électronique interactive de l'évangile selon St Jean [JON 03], mais ne peut servir à d'autres graphies.

Dans ce qui suit, après avoir exposé certaines spécificités graphiques du sanskrit, nous présentons les problèmes rencontrés pour créer un logiciel de génération assistée d'édition critique, en décrivant précisément les critères sur lesquels se fondent la comparaison des textes sanskrits. Puis, nous considérons l'application à ce contexte de techniques relevant de l'analyse de données que l'édition critique informatisée rend possible, en vue notamment de déterminer - au moins partiellement - la filiation des manuscrits à l'aide d'arbres phylogénétiques, ou encore à l'aide de méthodes de classification des manuscrits.

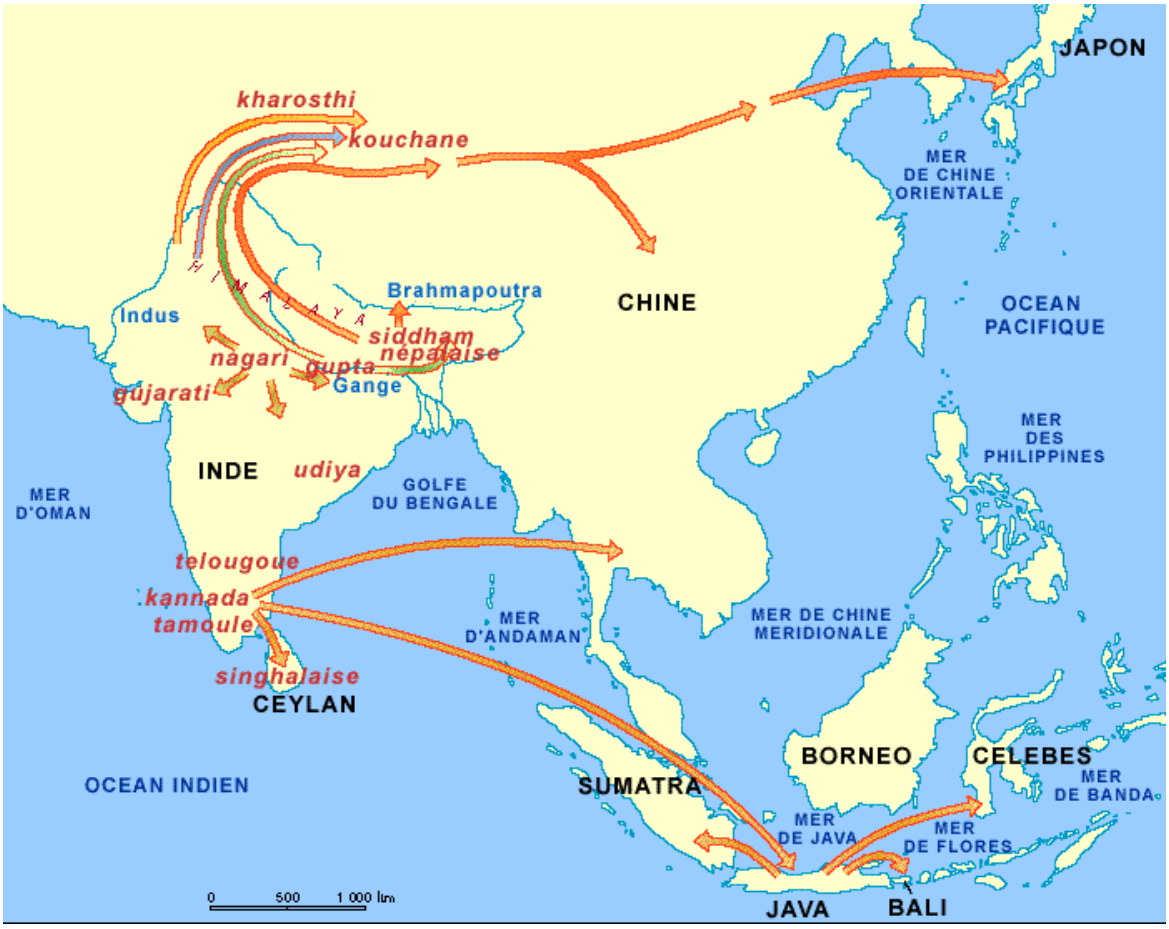


**Figure 1.** ***La transmission de la graphie du sanskrit.***

## 2. Le Sanskrit

Le Sanskrit fait partie des langues dites "indo-européennes" au même titre que le français. C'est une langue ancienne de l'Inde qui est devenue au cours des années non seulement une langue liturgique, mais aussi la "Lingua Franca" des lettrés indiens (les pandits). De nos jours encore, deux

pandits venant du nord et du sud de l'Inde, et ne parlant pas la même langue, vont converser en sanskrit s'ils évoquent des sujets de philosophie, de religion. Ils utiliseront plus naturellement l'anglais lorsqu'ils discuteront d'horaires d'avion ou des qualités d'une automobile ….

Deux phénomènes historiques différents ont influencé l'écriture du sanskrit. D'une part, du fait de l'extension de la culture indienne au cours des premiers siècles de notre ère dans toutes l'Asie du sud-est, un certain nombre de langues, généralement fort éloignées du sanskrit, ont adopté pour leur écriture la graphie du sanskrit ou un modèle dérivé[1].

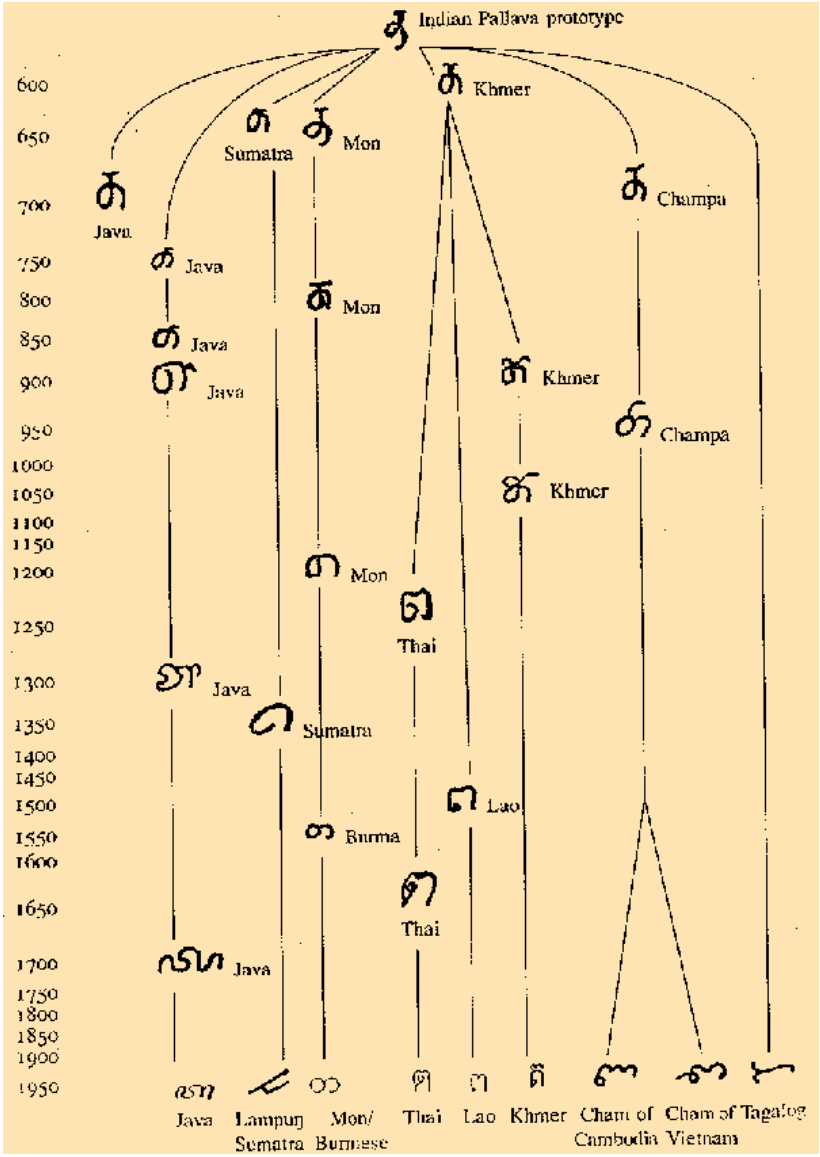


**Figure 2.** ***Langues à la graphie dérivée du sanskrit***

Ces langues ayant elles-mêmes parfois influencé leurs voisines, l'influence du sanskrit s'est ainsi propagée de manière indirecte. Parmi les vecteurs de cette diffusion, nous pouvons citer le khmer, le thaï, le birman, certaines langues lao … La carte de la figure 1 donne une idée de cette propagation qui s'est effectuée en Asie.

[1] De la même manière, les japonais ont adopté (et modifié) la graphie chinoise.

L’arbre décrit par la figure 2 permet de montrer les langages d'Asie du sud-est dont la graphie dérive du sanskrit. Toutes ces langues partagent avec celui-ci un assez grand nombre de propriétés, et par conséquent, peuvent être prises en compte par notre logiciel moyennant quelques adaptations.

Aux indes même, au cours du temps et selon les régions, la graphie du sanskrit s'est également modifiée. La figure 3 fournit un arbre phylogénétique de cette évolution. L'arbre reflète l'évolution du caractère ".n" (en translittération Velthuis) en fonction du temps et du lieu géographique. Ces deux figures se trouvent sur le site internet de Y. K. Malaiya [Mala 05]. Dans le même esprit, on peut trouver sur le site de LO [LO 05], un programme qui permet d'afficher différents caractères selon les graphies les plus courantes aux indes.

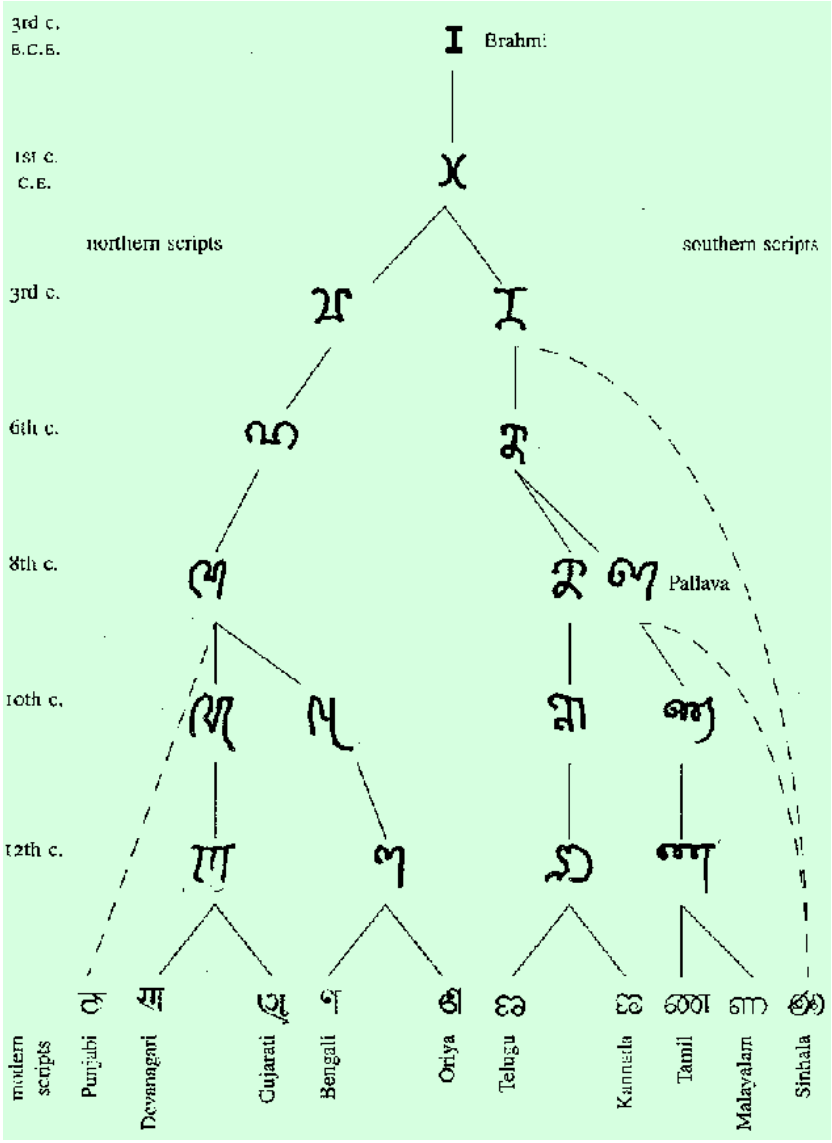


**Figure 3.** ***Différentes graphies sanskrites***

Un même texte sanskrit peut donc être écrit, selon le manuscrit considéré, en utilisant la graphie Nagari, Bengali ou Telugu. Seul un lecteur familier de la graphie utilisée pourra lire le texte, alors que le texte lui-même est compréhensible par tout sanskritiste.

## 2.1. La graphie du sanskrit

Considérons tout d'abord le texte du manuscrit sanskrit qui apparaît dans la figure 4 ci-dessous. Nous pouvons constater, sur cet exemple, deux caractéristiques graphiques élémentaires du sanskrit qui vont rendre plus difficile la création d'une édition critique. La première est que la graphie des lettres du Sanskrit ne nous est pas familière, ce n'est pas l'alphabet latin. La seconde est que les blancs sont éminemment rares dans ce texte, de longues séquences de lettres pouvant apparaître de manière quasi continue. Nous pouvons déjà imaginer l'effet de ce manque de séparation entre les mots sur la complexité des opérations de comparaison de deux manuscrits. Remarquons enfin que les lettres sont liées entre elles par une barre horizontale appelée "mantra" (la mère). La graphie utilisée dans ce manuscrit est la Devanagari.

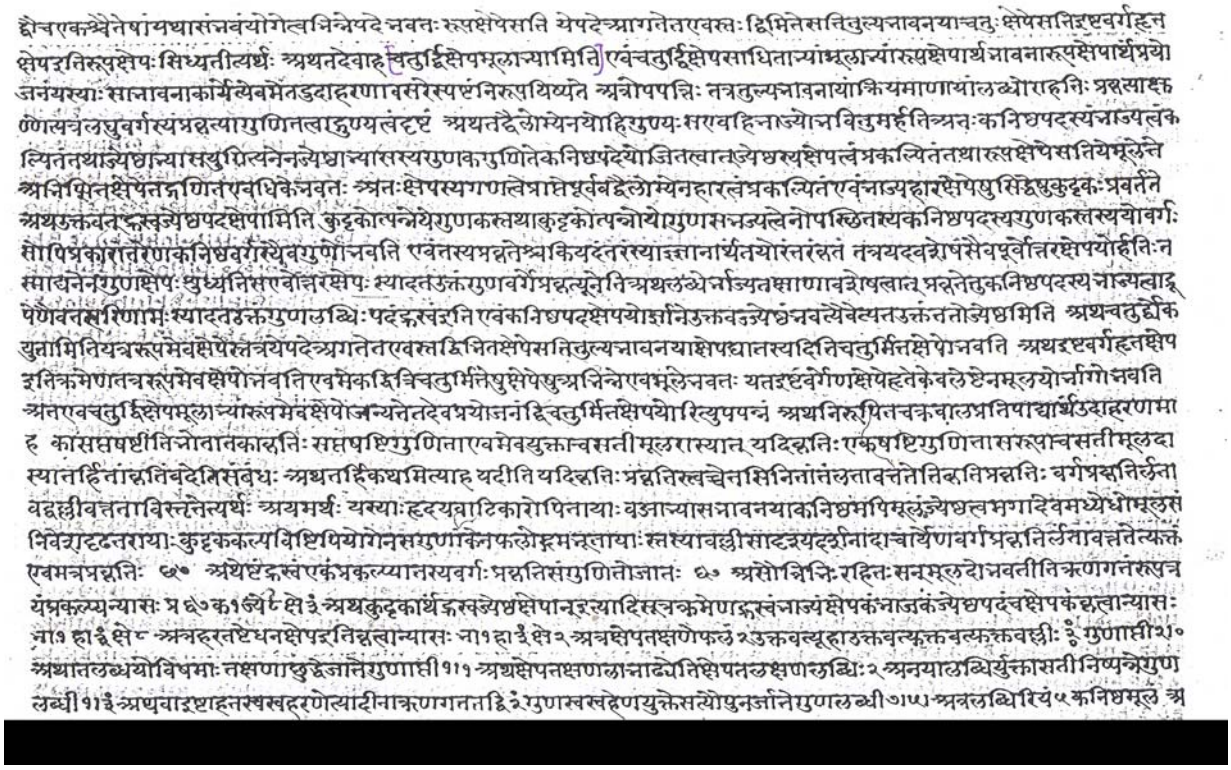

**Figure 4.** ***Un manuscrit Sanskrit***

Le sanskrit n'est pas la seule langue à pouvoir s'écrire sans espace entre les mots. Par exemple, les épigraphies latines qui ont été conservées, ne comportent pas d'espace et restent lisibles, mais sur une courte longueur. Bien souvent, les manuscrits écrits en minuscules carolingiennes, voire en onciales, ne comportent pas de blanc eux non plus, mais en revanche ils deviennent rapidement incompréhensibles passé le premier mot[2].

En ce qui concerne le sanskrit et les langues d'une graphie apparentée, l'absence de blanc rend dans certains cas, les textes ambigus, et c'est là parfois un effet de style recherché. Une

[2] Le site de la Bibliothèque Nationale de France [BNF] contient une description des débuts de l'écriture de notre langue, dont on pourra admirer les exemples présentés.

"désambiguïsation" automatique des textes dont la graphie dérive du sanskrit, a été étudiée par Del Vigna et Berment au sujet du Lao ([DEL 03]).

### 2.2. L'utilisation de code translitteré

La translittération consiste à écrire le sanskrit suivant l'alphabet latin, en faisant correspondre à chaque lettre sanskrit une séquence de lettres latines.

La translittération du sanskrit a d'abord été mise au point par l'administration britannique, à la suite des travaux de Sir William Jones, afin de pouvoir imprimer du sanskrit (voire de l'hindi) en se servant des fontes de caractères usuelles pour l'anglais (l'alphabet latin). Cette translittération qui possède son histoire propre, est approximativement conforme à la prononciation du sanskrit et est assez facilement compréhensible et utilisable pour un locuteur de cette langue. Ce type de translittération était néanmoins peu adapté à la saisie et l'affichage par ordinateur à l'aide des logiciels auxquels nous sommes habitués pour afficher les caractères ascii. Il a donc fallu créer de nouveaux systèmes de translittération adaptés à cette fin. Actuellement le plus courant est le système Velthuis, qui est directement dérivé de l'ancien système de translittération et qui est associé à un pré-processeur et à des fontes TeX conçues et dessinées par Franz Velthuis (cet ensemble, accompagné du mode d'emploi "ad hoc", est disponible sur toutes les archives TeX [TEXA]).

Une conséquence majeure de l'utilisation de la translittération est que les comparaisons de textes ne peuvent s'effectuer directement "lettre par lettre", celles-ci ne correspondant pas directement aux caractères latins, mais seulement "séquence par séquence", chaque séquence correspondant à la translittération latine d'un caractère sanskrit.

Comme nous le verrons au paragraphe 4, l'absence de caractère séparateur entre les mots augmente considérablement la complexité algorithmique. Afin de diminuer cette complexité et de rendre possible l'identification des mots sur lesquels porte la différence entre les textes, nous allons utiliser dans l'édition critique un texte lemmatisé, c'est-à-dire une version du texte incluant des signes spécifiques afin d'indiquer les séparations entre les mots, les racines, les préfixes, ... Cette version lemmatisée du texte de l'édition va servir de référence pour une comparaison avec tous les autres textes, et se nomme dans notre contexte : "padapatha", ce qui correspond à une forme de récitation sanskrite effectuée en détachant distinctement les mots et les syllabes. La forme non lemmatisée s'appelle un "samitapatha", c'est aussi le nom d'une forme de récitation du sanskrit qui est dite en respectant les liaisons entre les mots. Par ailleurs, mentionnons que le processus de lemmatisation est utilisé couramment comme préalable à un traitement plus sophistiqué d'un texte, notamment dans le domaine des statistiques textuelles (le lecteur intéressé par ce sujet pourra consulter le chapitre 2 du livre de L. Lebart et A. Salem [LEB 04]).

Enfin, on pourrait croire que la lemmatisation permet de résoudre l'essentiel des problèmes liés à la comparaison des textes sanskrits, mais il n'en est rien. En effet, les mots sanskrits se comportent comme des protéines : ils se transforment lorsqu'ils s'accolent. C'est ce que l'on appelle des Sandhi. Un médiocre exemple de sandhi est fourni en français par la transformation du préfixe privatif **in** en **im** devant un **m**, un **b** ou un **p** (par exemple, on écrit "**in**édit" mais "**im**muable"). Plus spécifiquement, une même séquence incluse à la fois dans le padapatha et dans un manuscrit, ne figure pas sous la même forme dans ces deux documents, en raison de la lemmatisation du padapatha et de l'existence de sandhi. Ainsi dans notre mauvais exemple français, il faudrait comparer "**in**^muable" avec "**im**muable". Remarquons enfin qu'il existe un "dernier plaisir" pour compliquer un peu les comparaisons : les multiples règles des copistes qui autorisent l'usage de

certains caractères à la place d'autres, ce qui donne lieu à des manuscrits qui diffèrent entre eux seulement en leurs formes …

### 2.3. L'affichage du sanskrit et Unicode

Dans ce qui suit, nous donnons un aperçu du contexte dans lequel ces problèmes se posent et nous esquissons la manière actuelle de les traiter. Si la possibilité d'imprimer à l'aide de l'informatique des textes écrits en sanskrit existe depuis longtemps grâce à TeX, l'affichage des caractères sanskrits sur le Web n'est devenu possible que grâce au standard Unicode.

Unicode, né du désir de trouver un standard d'affichage qui permette de prendre en compte les langues chinoise et japonaise, est un système de codage dont l'origine remonte aux années 1990. L'ambition d'Unicode [UNI] est de fournir un seul et unique système de codage qui prenne en compte l'ensemble des caractères existant de par le monde. Unicode nous permet de mêler dans un même texte, et de manière simple, des caractères chinois, des hiéroglyphes égyptiens, de l'anglais, du français et du grec ancien.

Malgré tous ses défauts, dont le principal est d'ignorer trop superbement la notion de glyphe, Unicode semble une tentative réussie. Chacun des caractères recensés par Unicode reçoit un code sur 21 bits, ce code étant naturellement unique. Les codages se font par ensemble relatif à une langue ou à un ensemble de langues. Par exemple, le codage LATIN-1 (équivalent au codage Iso-latin1) est couramment utilisé pour les langues de l'ouest européen, y compris pour le français mais à condition de lui adjoindre le codage "LATIN ETENDU A" qui permet d'afficher, entre autres, le caractère "œ" (cf. le mot cœur).

Tous les caractères Unicode ne sont pas logés à la même enseigne. Le codage d'un certain nombre de caractères commence par un nombre conséquent de zéro, ce qui permet lorsqu'aucune ambiguïté n'est à craindre, de les représenter dans un format beaucoup plus court que 21 bits. C'est le cas en particulier des caractères "ASCII" mais aussi des caractères indiens tels ceux de la graphie Devanagari.

Néanmoins, les caractères Unicode sont un peu comme les dieux du panthéon indien : ils apparaissent plus souvent sous la forme de leurs avatars que sous leur forme propre. De fait, nous ne "voyons" jamais de codes Unicode, mais uniquement leurs représentations. Il existe trois représentations majeures qui s'expriment par paquets de 8, 16 ou 32 bits. La forme la plus utilisée est la forme 8 bits, car elle permet d'afficher les caractères ASCII "tel quel" et procure, en moyenne, une meilleure compacité. Les caractères autres que ceux de l'Ascii sont codés en utilisant plusieurs éléments successifs de 8 bits. Pour la Devanagari qui nous concerne, il faut deux ou trois paquets de huit bits pour coder chaque caractère. Le lecteur curieux et intéressé par ces problèmes peut consulter les chapitres 2 à 4 du livre de Y. Haralambous : Fontes & Codages [HAL 04].

Une fois la transformation des caractères Velthuis en caractères Unicode effectuée, tous les problèmes d'affichage des caractères ne sont pas résolus pour autant. En effet, comme nous l'avons déjà mentionné, le sanskrit est une langue dans laquelle les ligatures (telles que la transformation du OE en Œ dans le mot cœur) abondent. Or, et c'est là un problème d'Unicode, coder les caractères et non pas les glyphes (image d'un ou plusieurs caractères) et cela sans les ligatures, comme le propose Unicode, conduit à un affichage du sanskrit qui est très difficilement lisible. Il faut donc trouver des polices de caractères (et les programmes capables de les utiliser), autorisant non seulement l'affichage des caractères sanskrits, mais permettant également d'effectuer les ligatures susceptibles d'être formées.

Nous pouvons voir ci-dessous le mot dont la translittération Velthuis est **KTHNA** apparaître dans l'image de gauche sans ligature, et dans celle de droite avec ligature : la différence est notable.

**sans ligatures** **avec ligatures**

**Figure 5.** ***Exemple de ligature***

## 3. La reconnaissance des séquences de caractères

D'après ce que nous avons pu dire précédemment sur la graphie du sanskrit, la comparaison des textes sanskrits, surtout quand elle s'effectue entre un padapatha (texte lemmatisé) et un samitapatha (texte non lemmatisé), ne peut se faire en utilisant les voies habituelles.

En effet les deux textes ne sont pas homogènes selon plusieurs critères :

- Le padapatha contient des caractères séparateurs (indiquant les frontières des lemmes) qui ne figurent pas dans le samitapatha.
- Du fait que le padapatha contient des mots séparés, les sandhi ne sont pas formés, alors qu'ils le sont dans le samitapatha.
- Les samitapatha contiennent des informations supplémentaires collectées par le pandit qui a effectué la saisie concernant le manuscrit dont il est issu. Ces informations peuvent être considérées comme des méta-données, elles concernent les ratures, les couleurs d'encre, le style d'écriture …
- Enfin, les deux textes peuvent contenir des commentaires des signes de ponctuation, une numérotation des vers dont il a été décidé de ne pas tenir compte pour la comparaison des manuscrits.

Une comparaison "caractère" par "caractère" surtout s'il s'agit des caractères latins de la translittération, ne peut manifestement pas convenir. Nous allons détailler ci dessous les étapes nécessaires.

Signalons auparavant un problème particulier : que faire lorsqu'apparaît dans un manuscrit une séquence qui n'est visiblement pas présente dans le texte maître ? Comme il n'existe pas de partie équivalente dans le padapatha, nous ne pouvons pas lemmatiser cette séquence, et donc savoir s'il s'agit d'un mot, d'un groupe de mots ou d'une phrase. La seule indication fiable dont nous pouvons disposer étant la longueur de ce texte supplémentaire, nous pouvons toutefois considérer raisonnablement qu'il s'agit d'un mot si la longueur de la séquence reste dans certaines limites. Si l'on dispose d'un lemmatiseur automatique de sanskrit, comme celui qui a été construit par Gérard Huet [Huet 05], alors on peut obtenir les formes lemmatisées les plus vraisemblables de la séquence en question, mais uniquement pour des phrases simples. Par conséquent, le cas de séquences additionnelles simples peut être entièrement résolu.

### 3.1. Les étapes du prétraitement

Afin de pouvoir effectuer nos comparaisons d'une manière homogène, un certain nombre de prétraitements doivent être effectués. Ils consistent en une série de traitements Lex[3] [LESK], ou plutôt son avatar Flex [PAX 95]. Dans ce qui suit, nous appellerons "séquence" toute suite de caractères sanskrits limitée par un blanc ou par un signe de ponctuation.

Les différents traitements Flex peuvent s'effectuer "au vol" et s'enchaîner l'un après l'autre, sans affecter le texte original. La liste de ces prétraitements est indiquée ci-après.

- Purger les commentaires, numéroter les paragraphes, garder le texte en mémoire ;
- Pour le Padapatha :
    - Faire apparaître les sandhi "ajouteurs" de blancs qui peuvent faire apparaître de nouvelles séquences ;
- Traiter et faire disparaître les "méta-données" du samitaptha du texte de la comparaison ;
- Faire apparaître les séquences du texte en tenant compte des séquences ajoutées pour le padapatha ;
- Pour le padapatha :
    - Faire apparaître les Sandhi du padapatha ;
- Reconnaître les caractères Velthuis.

Lorsque tous ces prétraitements ont été réalisés, la comparaison des textes sanskrits peut alors commencer, caractère par caractère.

### 3.2. Les expressions régulières et l'utilisation de FLEX

Nous n'avons pas l'intention de donner ici une définition précise et détaillée de la théorie des langages, et par conséquent, les définitions manquantes resteront dans le flou de la langue commune.

La notion d'expression régulière peut être définie de la manière suivante :

Soit $V_t$ un ensemble de lettres, appelé vocabulaire terminal ou alphabet, que l'on suppose muni d'une opération de concaténation notée parfois "**.**" (ou non notée s'il n'y a pas d'ambiguïté), d'une opération d'alternance, appelé "ou" et noté "|", et enfin d'un opérateur de répétition, noté *, appelé aussi "étoile de Kleene".

Les éléments de la clôture de l'ensemble $V_t$, par ces trois opérateurs, sont appelés "expressions régulières". En d'autres termes, les expressions régulières se définissent par récurrence à l'aide des deux règles suivantes :

- Toute lettre appartenant à $V_t$ est une expression régulière ;

[3] Lex *est un générateur d'analyseurs lexicaux.*

- Si r et r' sont deux expressions régulières, alors r.r', r|r' et r* sont des expressions régulières.

Par exemple, si $V_t$ désigne l'alphabet latin, alors :

*toto* est une expression régulière, obtenue par exemple par la concaténation **t.o.t.o** ;

*blanc|bleu* est une expression régulière dont l'automate associé est capable de reconnaître les chaînes qui sont soit la chaîne **bleue,** soit la chaîne **blanc** ;

*c(a|b)** est une expression régulière dont l'automate associé est capable de reconnaître le caractère *c* suivi d'une suite pouvant être vide de a ou de b, par exemple : **cabaaba, caaaa, cbbb, caaabbb, c.**

Un Automate à Etats Finis ou AEF (Automates à nombre Fini d'Etats, serait plus précis) est une machine mathématique qui lit les chaînes de caractères et qui répond par OUI ou NON selon que la chaîne lue appartient ou non au langage qu'elle reconnaît.

Le programme FLEX permet de générer des Automates à Etats Finis à partir d'un ensemble d'expressions régulières. Le processus se schématise de la manière suivante :

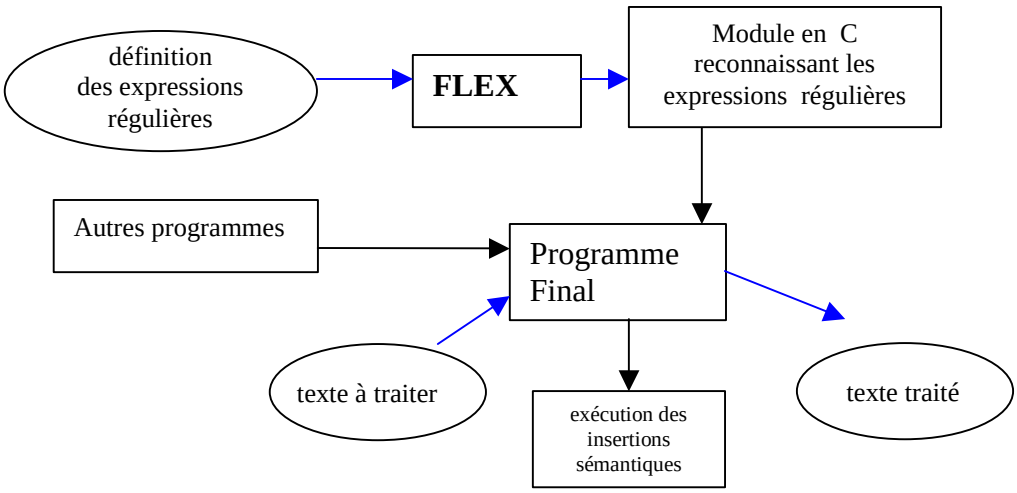


**Figure 6.** ***Le principe de fonctionnement de Flex***

Le module en langage C généré par FLEX peut être associé à n'importe quel autre module de manière à former un programme. Chacun des automates reconnaît une des expressions régulières données, et peut associer à cette expression une séquence d'instructions C spécifique "insertion sémantique".

Dans FLEX, les expressions régulières peuvent être reconnues au sein de deux contextes : le contexte gauche et le contexte droit. Nous ne considérons ici que le contexte droit qui est le plus simple à utiliser. Ce contexte est indiqué sous FLEX par le signe "/" . Ainsi l'expression régulière

*abc* reconnaît **abc** partout dans un texte, alors que l'expression régulière *abc/def* reconnaît **abc** dans un texte uniquement quand il est suivi de **def**.

Les expressions régulières peuvent être utilisées dans FLEX à de nombreuses fins, mais en ce qui nous concerne, nous les utilisons essentiellement dans 3 buts :

- garder le texte reconnu par les expressions en ignorant le texte non reconnu,
- ignorer le texte reconnu par les expressions en gardant le texte non reconnu,
- coder des expressions reconnues.

Nous utilisons le codage à la fois pour coder les caractères de l'alphabet Velthuis et pour effectuer certaines transformations de sandhi en nous servant du contexte droit qui est parfaitement adapté à la reconnaissance des sandhi.

Pour finir, voici une série d'exemples d'expressions régulières, légèrement simplifiées pour des raisons de lisibilité. Les éléments compris entre accolades sont des éléments prédéfinis comme `SPECIAL` qui désigne un caractère spécial utilisé pour la lemmatisation, et `VOY` qui désigne une voyelle quelconque.

*Premier exemple :*

- `a` `return LettreAc;`
- `aa` `return LettreAL;`

Ces deux expressions peuvent s'interpréter de la manière suivante. Si nous rencontrons un caractère **a** seul (a court) nous retournons le code LettreAc; si nous rencontrons **aa** (a long) nous retournons le code LettreAL. Les éléments reconnus sont retirés de la chaîne traitée.

*Second exemple :*

- `(a|aa){SPECIAL}(a|aa)` `return LettreAL;`

Cette expression indique que si l'on rencontre un **a** (a court) ou un **aa** (a long) suivi d'un caractère de lemmatisation puis d'un autre a (court ou long), nous devons retourner dans tous les cas le code de la lettre **aa** (a long) : LettreAL.

*Troisième exemple :*

- `(i|ii)/{SPECIAL}{VOY}` `return LettreY;`

Cette expression indique que si l'on rencontre un **i** (i court) ou un **ii** (i long) suivi d'un caractère de lemmatisation puis d'une voyelle, nous devons retourner de la lettre y (LettreY). Seul le **i** (court ou long) fait partie de la chaîne reconnue, et est retiré du texte traité. Le caractère spécial et la voyelle qui suivent, font partie du contexte droit et ne sont pas retirés du texte traité.

## 4. Comparaison de textes sanskrits : élaboration d'une distance

La comparaison des textes que nous présentons dans ce papier, comporte deux objectifs :

1) Déterminer les différences ‘qualitatives’ qui existent entre les textes des manuscrits en vue de l'élaboration d'une édition critique. Le résultat de cette opération doit indiquer les mots ou les phrases qui diffèrent entre chaque manuscrit et le texte maître.

2) Evaluer les dissimilarités entre les manuscrits en vue d'établir des relations de filiation, voire des classifications entre eux.

La principale difficulté réside dans la structure même du sanskrit, en particulier, elle provient de l'absence possible de blanc entre deux mots consécutifs, ce qui rend la notion de mot extrêmement imprécise. Pour illustrer ce point, revenons à un exemple français : le texte "lepetitchatestmort" est clair pour toute personne qui le lit, mais ne permet pas à un programme démuni de lexique de distinguer les mots entre eux. La même difficulté apparaît en ce qui concerne les manuscrits sanskrits. Comment déterminer en comparant deux chaînes de caractères, les mots qui diffèrent, et plus généralement, quels sont les mots ?

En fait, pour comparer les textes, nous n’avons pour l’instant que la possibilité de confronter chaque manuscrit au padapatha qui est le seul texte lemmatisé dont nous disposons. Par conséquent, en ce qui concerne notre premier objectif, nous procédons essentiellement par alignement de deux séquences, l’une provenant d’un manuscrit (sam et l’autre du texte maître le padapatha. Pour effectuer chaque alignement, nous nous baserons sur les distances d’édition comme cela se pratique en biologie moléculaire pour les comparaisons de longues molécules, et plus particulièrement, sur les distances fournies par la méthode dite de plus Longue Séquence Commune (Longuest Common Subsequence), notée LCS. Plus précisément, nous utiliserons la solution donnée par un des algorithmes les plus connus, l’algorithme DIFF d’Unix [HUN 77]. Notons toutefois qu’au vu des spécificités des manuscrits sanskrits, l’information fournie par un seul alignement sera généralement insuffisante pour déterminer toutes les différences recherchées. Par ailleurs, la comparaison de deux manuscrits ne peut être qu’indirecte, puisqu’ils sont tous les deux comparés uniquement au padapatha.

Pour réaliser le deuxième objectif, c’est-à-dire établir une distance entre les manuscrits, la même difficulté se présente. Si on se limite à la simple comparaison de chacun des manuscrits au Padapatha, une représentation fidèle des distances ainsi obtenues, conduit nécessairement à une forme en étoile dont le centre serait le padapatha, et il est clair que ce type d’analyse ne peut conduire qu’à des conclusions dont l’intérêt reste limité …

### 4.1. Les alignements de séquences

Comme nous venons de le mentionner, l’outil de base algorithmique pour réaliser les alignements est la méthode LCS (*i.e.*, la détermination d’une plus Longue Séquence Commune entre les deux parties de texte comparées). La recherche d'un simple alignement n'est pourtant pas suffisante pour permettre d'effectuer complètement les comparaisons, car il reste encore des problèmes annexes liés à la graphie du sanskrit. Avant de continuer plus avant, et essentiellement pour des raisons de terminologie, nous présentons maintenant la structure en chapitres et paragraphes des textes à comparer :

- Les textes à comparer sont divisés en chapitres, chaque chapitre d'un manuscrit étant représenté par un fichier.
- Chaque chapitre est divisé en paragraphes, et chaque paragraphe est porteur d'un numéro. La numérotation des paragraphes est unique, c'est-à-dire que le numéro d'un

paragraphe dans un manuscrit ne change pas dans tous les autres manuscrits où il est présent. De plus, le padapatha contient tous les numéros de paragraphes possibles dans un chapitre.

- Chaque paragraphe est divisé en séquences, une séquence désignant une suite de caractères séparés par un blanc ou un signe de ponctuation (la ponctuation peut contenir des indications propres à la versification).

  Comme les blancs sont parfois placés "au hasard", la notion de séquence est imprécise en sanskrit. Elle est néanmoins très utile au profane pour pouvoir se repérer au milieu d'un texte qui lui apparaît de prime abord comme un simple amoncellement de caractères.

Les comparaisons vont donc s'effectuer, sans aucune ambiguïté, chapitre par chapitre, paragraphe par paragraphe, puisque chapitres et paragraphes sont chacun identifiés par un nom ou par un numéro. Les comparaisons de séquences seront moins simples, et ceci pour deux raisons :

- Comme nous l'avons déjà mentionné les caractères "blanc" séparateurs de séquences peuvent être disposés non pas au hasard, mais au gré de l'inspiration d'un scribe, cette inspiration pouvant varier au cours des époques. Nous pourrions aisément trouver deux textes identiques mais composés de séquences différentes.
- L'ordre des mots n'est pas en sanskrit un élément important : deux phrases peuvent être considérées comme identiques alors qu'elles ne le seraient pas en français. Pour paraphraser le maître de philosophie du bourgeois gentilhomme [POQ 98] les phrases: "me font mourir d'amour" et "d'amour mourir me font" sont considérées en sanskrit comme absolument égales.

Le premier point peut être résolu par une méthode assez inattendue, que nous avons utilisée sans pour l'instant l'intégrer au reste de nos programmes, et c'est à notre grande surprise que nous avons découvert ses excellents résultats. Il s'agit de la méthode de Gale et Church [GALE 93]. L'idée d'utiliser cette méthode revient à M. Le Pouliquen au cours de son stage de Mastère.

## 4.2. La Méthode de Gale & Church

La méthode de Gale & Church [GALE 93] a pour but d'aligner des phrases sémantiquement correspondantes dans un corpus bilingue. Elle utilise la longueur des phrases comme critère d'alignement. Bien que ce critère soit extrêmement simple, il donne de très bons résultats pour des langues relativement proches telles le français, l'anglais et l'allemand. Ces langues ont en effet une structure de texte comparable. L'algorithme procède dans son principe de la manière suivante. Tout d'abord, les deux textes à comparer sont mis en regard en alignant les paragraphes qui se correspondent d'un texte à l'autre. Cette étape ne présente pas de difficulté particulière, bien que de (rares) erreurs soient toujours possibles. Puis les phrases sont alignées à l'intérieur de chaque paragraphe. L'algorithme utilise le fait que des phrases longues (respectivement courtes) auront plutôt tendance à être traduites par des phrases longues (respectivement courtes). Un score probabiliste, basé entre autres sur le rapport des longueurs des deux phrases, est mesuré pour chaque paire de phrases examinée. Une technique de programmation dynamique est utilisée afin de déterminer la paire de phrases qui maximise le maximum de vraisemblance d'un alignement de deux phrases. Notons que pour chaque paragraphe, les premières (respectivement dernières) phrases des deux textes sont nécessairement alignées. Remarquons enfin que l'algorithme ne

recherche pas seulement les coïncidences entre deux phrases, mais aussi entre une phrase et deux autres consécutives, voire entre deux phrases consécutives et deux autres également consécutives.

Bien que les deux textes à comparer soient, dans notre cas, écrits chacun en sanskrit, leur ensemble s'apparente à un texte bilingue. D'une part, le padapatha contient des signes de lemmatisation absents du samitapatha, et d'autre part le samitapatha contient des annotations en anglais écrites par le pandit qui effectue la saisie, et qui peuvent allonger le texte de manière notable. Les résultats de l'algorithme de Gale & Church sont tout à fait satisfaisants au niveau des séquences (texte séparé par un blanc ou un signe de ponctuation).

La figure 7 contient les résultats que fournit cet algorithme. Chaque ligne du samitapatha est précédée du symbole >> , et chaque ligne du padapatha est précédé de <<. On peut constater que les alignements se font convenablement, même lorsque du texte anglais se mêle au sanskrit et que deux lignes du samitapatha correspondent à une seule ligne du padapatha.

```
>> .r .l ityetau var.naavupadi"sya puurvaa.m"scaante kakaaramita.m karoti
<<.r .l iti+etau var.nau+upa^di"sya puurvaan+ca+ante ka_kaaram+itam+karoti

>> pratyaahaaraartham| tasya graha.na.m bhavati
>> \afterc{tribhi.h aka.h savar.ne diirgha.h}
<<prati^aa^haara_artham| tasya graha.nam+bhavati tribhis+| akas+sa_var.ne

>>\beforec{ tribhiraka.h savar.ne diirgha.h}
>>(6..1..101) ityakaare.na, iko gu.nav.rddhii (1..1..3)
<<diirgha.h (6..1..101) iti+a_kaare.na, ikas+gu.na_v.rddhii (1..1..3)

>>\afterc{itiikarina}\beforec{itiikaare.na}, ugita"sca (4..1..6) ityukaare.na| akaaraadayovar.naa.h
<<iti+i_kaare.na, uk_itas+ca (4..1..6) iti+u_kaare.na| a_kaara_aadayas+var.naas+

>>pracuraprayogavi.sayaaste.saa.m suj~naanamupade"se prayojanam|
<<pra^cura_pra^yoga_vi.sayaas+te.saam+su_j~naanam+upa^de"se pra^yojanam|

>>.lkaarastu k.lpistha eva prayujyate| k.lpe"sca puurvatraasiddham
<<.l_kaaras+tu k.lpi_sthas+eva pra^yujyate| k.lpes+ca puurvatra+a_siddham

>>(8..2..1) iti \afterc{sa} \beforec{la}tvam \afterc{prasiddham tatra .rkara eva} \beforec{asiddham, tasyaasiddhatvaad.rkaara eva} ackaaryaa.ni
 <<(8..2..1) iti latvam+a_siddham, tasya+a_siddhatvaat .r_kaare+eva+ac_kaaryaa.ni

>>bhavi.syantiiti kimartham .lkaara upadi"syate? latvavidhaanaadyaani
<<bhavi.syanti+iti kim_artham .l_kaaras+upa^di"syate? latva_vi^dhaanaat+yaani

>>paraa.nyackaaryaa.ni taani .lkaare yathaa syuriti| kaani punastaani|
<<paraa.ni+ac_kaaryaa.ni taani .l_kaare yathaa syus+iti| kaani punar+taani|

>>pluta.h , svarita.h , dvirvacana.m| k.l3\afterc{pra} \beforec{pta}"sikha.h , prak.lpta,
<<plutas+, svaritas+, dvis_vacanam| k.l3pta_"sikhas+, pra_k.lptas+,
```

**Figure 7.** ***Résultats de l'algorithme de Gale-Church***

### 4.3. Les alignements via les LCS

Les alignements via la méthode LCS relèvent du cadre général du calcul des distances d'édition dont le prototype est la distance de Levenshtein. Rappelons que la distance de Levenshtein a pour but de déterminer le coût total minimal des opérations nécessaires pour transformer une chaîne de caractères en une autre, ce coût total étant égal à la somme des coûts des opérations élémentaires. Les opérations élémentaires autorisées sont au nombre de trois : ajout, suppression et substitution. Considérons l'exemple suivant qui consiste à comparer les chaînes ATGCTA et ACGA, en supposant que toutes les opérations élémentaires autorisées possèdent un coût unitaire égal à 1.

Le tableau 1 ci-dessous montre le calcul de la distance de Levenshtein entre les chaînes ATGCTA et ACGA, notées respectivement CH1 et CH2. Ce calcul est effectué selon la méthode de la programmation dynamique.

Plus précisément, chaque case t[i,j] du tableau 1 contient la distance d'édition entre les i premières lettre de CH1 et les j premières lettres de CH2, c'est-à-dire indique le nombre (minimum) d'opérations nécessaires pour passer d'une chaîne à l'autre. Le tableau se remplit en partant de la case du coin "haut gauche". Dans notre exemple, cette case correspond à la distance entre le premier caractère de CH1 et le premier caractère de CH2, soit entre "A"et "A".

Les cases suivantes se remplissent par récurrence, plus précisément en appliquant les règles suivantes :

$$t[1,j+1] = t[1,j] + \text{ajout}(CH1[j+1]) ;$$

$$t[i+1,1] = t[i,1] + \text{ajout}(CH2[i+1]) ;$$

$$t[i+1,j+1] = \min(t[i,j] + \text{subs}(CH1[j+1],CH2[i+1]),$$
$$t[i+1,j] + \text{ajout}(CH1[j+1]),$$
$$t[i,j+1] + \text{ajout}(CH2[i+1])).$$

Dans notre exemple, nous avons :

$t[1,2] = t[1,1] + 1$ ; $t[2,1] = t[1,1] + 1$ et $t[2,2] = t[1,1]+1$.

La valeur qui se trouve dans le coin "bas droit" d'un tel tableau correspond à la distance entre les deux chaînes comparées, et il en est de même pour chaque cellule du tableau qui contient la distance entre les deux sous-chaînes associées à la cellule considérée. Le chemin qui indique les opérations nécessaires pour transformer une chaîne en l'autre, est celui qui part du coin "bas droit" et rejoint le coin "haut gauche" en choisissant à chaque fois la plus petite des trois valeurs possibles. Lorsqu'il existe plusieurs plus petites valeurs possibles, toutes les alternatives sont explorées. Les différents chemins ainsi définis qui conduisent au coin "haut gauche", définissent les séquences d'opérations nécessaires pour transformer une chaîne en une autre, avec un coût minimal. De plus, chaque chemin peut être représenté par un alignement entre les deux chaînes comparées. Dans le cas de l'exemple de la comparaison des chaînes CH1 = ATGCTA et CH2 = ACGA, on obtient le tableau 1 ci-dessous.

| | | A | T | G | C | T | A |
|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| A | 1 | 0 | 1 | 2 | 3 | 4 | 5 |
| C | 2 | 1 | 1 | 2 | 2 | 3 | 4 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| G | 3 | 2 | 2 | 1 | 2 | 3 | 4 |
| A | 4 | 3 | 3 | 2 | 2 | 3 | 3 |

**Tableau 1.** ***Distance de Levenshtein***

Dans cet exemple, il existe deux chemins ayant un coût minimal. Ils sont indiqués dans le tableau par la mise en valeur des cases qu'ils traversent. Les deux chemins peuvent être définis, de façon équivalente, à l'aide de deux alignements représentés ci-dessous :

```
A T G C T A          A T G C T A
A - - C G A          A C G - - A
```

Chaque alignement s'interprète de la façon suivante :

- une paire alignée du type $\begin{pmatrix} A \\ B \end{pmatrix}$ indique la substitution de la lettre A par la lettre B,
- une paire alignée du type $\begin{pmatrix} A \\ - \end{pmatrix}$ indique la suppression de la lettre A,
- et une paire alignée du type $\begin{pmatrix} - \\ B \end{pmatrix}$ indique l'insertion de la lettre B.

Lorsque l'on donne à l'opération de substitution un coût supérieur à celui d'un ajout suivi d'une suppression, alors l'algorithme devient celui de la recherche d'une plus longue séquence commune (LCS). Si l'on utilise cet algorithme, la longueur d'une plus longue séquence commune est la valeur que l'on trouve dans le coin "bas droit" du tableau – l'interprétation des autres cases du tableau est identique : la case (i,j) contient la plus longue séquence commune entre les deux sous-chaînes obtenues en extrayant respectivement les i et les j premiers caractères des deux chaînes comparées.

Considérons l'exemple suivant (extrait de Charras et Lecroq [CHA]) qui consiste à déterminer l'ensemble des plus longues sous-séquences communes entre AGCGA et CAGATAGAG.

| | | C | A | G | A | T | A | G | A | G |
|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| A | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| G | 0 | 0 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| C | 0 | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| G | 0 | 1 | 1 | 2 | 2 | 2 | 2 | 3 | 3 | 3 |
| A | 0 | 1 | 2 | 3 | 3 | 3 | 3 | 3 | 4 | 4 |

**Tableau 2.** ***Calcul d'une plus Longue Séquence Commune***

Ce tableau permet d'obtenir les quatre alignements optimaux suivants, tous relatifs à la même plus longue séquence commune AGGA :

```
- A G C - - - G A -
C A G - A T A G A G

- A G - - C - G A -
C A G A T - A G A G

- A G - C - - G A -
C A G A - T A G A G

- A G - - - C G A -
C A G A T A - G A G
```

### 4.4. La construction des distances entre les manuscrits

La comparaison de deux manuscrits s'effectue sur la base d'un découpage en mots des deux textes qui est obtenu après avoir comparé chacun d'eux avec le texte lemmatisé "padapatha". La procédure de comparaison s'effectue séquentiellement selon l'ordre des mots du "padapatha". Les différences sont de trois types : celles qui apparaissent entre deux mots, celles qui sont relatives aux assemblages de mots (tournures de phrases) et qui ressemblent à la différence existant entre "d'amour mourir me font" en lieu et place de "me font mourir d'amour", et enfin celles qui concernent les pans de phrase rajoutés ou oubliés par un scribe. L'identification de ces pans de phrases, consiste à déterminer les endroits où s'arrêtent les parties de texte oubliées ou rajoutées, ce qui entraîne une importante difficulté algorithmique.

Pour comparer deux mots, nous calculons la longueur d'une plus longue séquence commune (LCS) à l'aide de l'approche présentée au paragraphe précédent. Plus précisément, nous proposons de calculer la distance entre deux mots x et y en utilisant la formule suivante :

$$d(x,y) = \frac{|x| + |y| - 2 * LCS(x,y)}{|x| + |y|},$$

où LCS(x,y) désigne la longueur d'une plus longue séquence commune entre x et y.
Le tableau 3 illustre ce type de comparaison entre "mourir" et "d'amour ". Ce tableau se lit de la manière suivante : la première ligne contient les indices de colonne correspondant aux caractères du mot "d'amour". La première colonne comprend les mêmes informations pour le mot "mourir". La seconde ligne contient le premier des mots à comparer M1 = "d'amour", la seconde colonne, le second mot à comparer : M2 = "mourir". Chaque case du tableau d'indice (n,m) contient le nombre de caractères communs entre les n premiers caractères de M1 et les m premiers caractères de M2. Le tableau est bâti suivant l'algorithme de la programmation dynamique. Le coin inférieur droit du tableau contient le nombre de caractères d'une plus longue séquence commune.

| | | *-1* | *0* | *1* | *2* | *3* | *4* | *5* | *6* |
|---|---|---|---|---|---|---|---|---|---|
| | | | ***D*** | ***'*** | ***A*** | ***M*** | ***O*** | ***U*** | ***R*** |
| *-1* | | *0* | *0* | *0* | *0* | *0* | *0* | *0* | *0* |
| *0* | ***M*** | *0* | *0* | *0* | *0* | *1* | *1* | *1* | *1* |
| *1* | ***O*** | *0* | *0* | *0* | *0* | *1* | *2* | *2* | *2* |
| *2* | ***U*** | *0* | *0* | *0* | *0* | *1* | *2* | *3* | *3* |
| *3* | ***R*** | *0* | *0* | *0* | *0* | *1* | *2* | *3* | *4* |
| *4* | ***I*** | *0* | *0* | *0* | *0* | *1* | *2* | *3* | *4* |
| *5* | ***R*** | *0* | *0* | *0* | *0* | *1* | *2* | *3* | *4* |

**Tableau 3.** ***Chemins LCS entre MOURIR et D'AMOUR.***

Dans cet exemple, il existe une seule plus longue séquence commune, et les deux chemins indiqués, correspondant aux deux alignements optimaux, sont :

```
- - - M O U R I R          - - - M O U R I R
D ' A M O U R - -          D ' A M O U - - R
```

La longueur de cette plus longue séquence commune étant égale 4, la distance entre mourir et d'amour est (6+7-2*4) / (6+7) = 5/13 = 0.385.

En ce qui concerne les différences entre tournures de phrase, nous avons adapté l'algorithme DIFF sans augmenter sa complexité algorithmique. Notre approche implique évidement d'avoir défini dans les deux textes comparés, une unité lexicale appelée MOT.

Nous proposons de reconnaître les phrases "équivalentes" avec un algorithme qui consiste d'abord à calculer les distances entre chaque paire de mots issus des deux phrases. Puis, l'algorithme calcule la distance entre les deux phrases comme étant le poids minimum d'un chemin hamiltonien dans le graphe biparti complet qui est défini de la manière suivante : chaque partie du graphe biparti représente une phrase, chaque sommet un mot d'une des deux phrases, et chaque arête qui relie donc deux mots des deux phrases, est valuée par la distance entre ces deux mots. En transposant au français, il faut pouvoir retrouver "d'amour mourir me font" en lieu et place de "me font mourir d'amour", pour paraphraser le bourgeois gentilhomme [POQ 98]. Le tableau 4 permet de calculer la distance selon cette méthode entre les deux phrases. Nous pouvons constater que chaque ligne contient une distance égale à 0, et par conséquent, il existe un hamiltonien de poids nul : les deux phrases sont équivalentes.

| | me | font | mourir | d'amour |
|---|---|---|---|---|
| d'amour | 0.778 | 0.818 | 0.385 | 0.000 |
| mourir | 0.750 | 0.800 | 0.000 | 0.385 |
| me | 0.000 | 1.000 | 0.750 | 0.778 |
| font | 1.000 | 0.000 | 0.800 | 0.818 |

**Tableau 4.** ***Distance entre deux phrases***

Il apparaît donc que la définition de notre dissimilarité intertextuelle dépend de plusieurs facteurs : distances entre mots, distances entre phrases, dissimilarités entre paragraphes que nous n'avons fait qu'évoquer (pans de phrases oubliés ...), ainsi que des dissimilarités générées par les méta-données accessibles (couleur de l'encre, style de graphie, ...) qui est un autre aspect également évoqué.

## 5. Conclusion

La masse considérable d'informations traitées par les (récents) logiciels d'édition critique (cf. [OHA 93], [ROB 94,00], [LAV 96] et [ANA 00]) conduit à diverses questions d'analyse de données exploratoire qui sont souvent spécifiques de l'œuvre étudiée et du langage utilisé. Il est plus facile de répondre à ces questions à l'aide d'un logiciel offrant des possibilités de visualisation ([MON 02]). Dans ce texte, nous avons présenté les nombreuses spécificités du sanskrit et les contraintes informatiques qu'elles engendrent, puis nous avons introduit des distances et des dissimilarités entre textes qui mesurent chacune l'écart entre deux textes en tenant compte de ces spécificités. Par la suite, nous envisageons de construire des hypothèses de filiation des manuscrits,

à l'aide notamment de représentations arborées ([BUN 71], [BAR 91]) basées sur une dissimilarité "synthétique" obtenue comme moyenne pondérée des dissimilarités relatives à un critère particulier, le choix de cette pondération pouvant être guidé par l'exigence de stabilité de la représentation arborée qui en résulte.

## Remerciements

*Les auteurs remercient le CNRS pour son soutien dans le cadre de l'ACI 'Histoire des savoirs', ainsi que la communauté européenne qui leur a permis de poursuivre leurs travaux de recherche dans de bonnes conditions grâce à un contrat ITT Asia.*

## REFERENCES

[ANA 00] Scholarly Digital Editions Leicester (UK)
URL : http://server30087.uk2net.com/hengwrt

[BAR 91] BARTHELEMY J.-P. & GUENOCHE A. (1991) *Trees and Proximity Representations*, John Wiley & Sons (première édition française : *Les arbres et les représentations des proximités*, Paris : Masson 1988).

[BNF] Bibliothèque Nationale de France : *Ecritures grecque et latine.*
URL : http://classes.bnf.fr/dossiecr/sp-voye3.htm

[BUN 71] BUNEMAN P. (1971) *Filiations of Manuscripts* Mathematics in Archaeological and Historical Sciences Edinburgh University Press.

[CHA] CHARRAS C. & LECROQ T. *Sequence Comparison*, animation à l'aide d'appliquettes Java d'algorithmes classiques d'alignements de séquences par programmation dynamique.
URL : http://www-igm.univ_mlv.fr/~lecroq/seqcomp

[CRO 01] CROCHEMORE, HANCART & LECROCQ. (2001) *Algorithmique du texte*, Vuibert, Paris.

[DEL 03] DEL VIGNA C. et BERMENT V. (2003) *Ambiguïtés irréductibles dans les monoïdes de mots,* Actes des 9èmes journées montoises d'informatique théorique, Montpellier, Sept 2002.

[HALA 04] Haralambous Y. (2004) *Fontes et Codage,* Editions O'reilly, Paris.

[HUN 77] HUNT J.W. & SZYMANSKI T.G. A fast algorithm for computing longest common subsequence CACM 20:5 1977.

[HUE] Huet G., *Parsing assistant for simple phrases.*
URL : http://sanskrit.inria.fr/DICO/reader.html

[GALE 93]Gale W.A., Church K.W. *A program for Aligning Sentences in Bilingual Corpora*. Computational Linguistic 19:1 75-102, 1993.

[JON 03] Westfälische Wilhelms-Universität MünsterSchlossplatz 2 48149 Münster
URL : http://nestlealand.uni-muenster.de/AnaServer?NAtranscripts+0+start.anv

[LAV 96] John Lavagnino & Dominik Wujastyk. Critical Edition Typesetting: The EDMAC format for plain TeX (San Francisco and Birmingham: TeX Users Group 1996). 108 pages, ill.

[LEB 04] Lebart L. & Salem A. (2004) *Statistique textuelle*, Dunod, Paris.

[LESK ] M. E. Lesk and E. Schmidt, `LEX--Lexical Analyzer Generator', *Unix Research System Programmer's Manual*, Tenth Edition, Volume 2

[LO 05] Lo Lawrence K. *Ancient Script*
URL : http://www.ancientscripts.com/sa_ws_cmp.html

[MAL 05] Malaiya Y. K. *Languages and Scripts of India.*
URL : http://www.cs.colostate.edu/~malaiya/scripts.html

[MON 02] Monroy C., Kochumann R., Furuta R, Uribina E., Melgoza E., Goenka A.: *Visualization of Variants in Textual Collations to Analyse the Evolution of Literary Works in the Cervantes Project.* Proceedings of the 6th European Conference, ECDL 2002. (Rome, Italy, September 2002). Maristella Agosti and Constantino Thanos, eds. Berlin: Springer, 2002. 638-53.

[OHA 93] O'Hara, Robert J., and Peter M.W. Robinson. 1993. *Computer-assisted methods of stemmatic analysis.* Occasional Papers of the Canterbury Tales Project, 1: 53–74. (Publication 5, Office for Humanities Communication, Oxford University.)

[PAX 95] Paxson V. *Flex A fast scanner generator*.
URL : http://www.gnu.org/software/flex/manual/

[POQ 98] Poquelin J.B. dit Molière *Le Bourgeois Gentilhomme Acte II, Sc 4* Larousse: Petits classiques Larousse 10 juillet 1998

[REN 96] RENOU L. (1996) *Grammaire sanskrite : phonétique, composition, dérivation, le nom, le verbe, la phrase*. Maisonneuve réimpression, Paris.

[ROB 94] Robinson, P., 'Collate: *A Program for Interactive Collation of Large Textual Traditions'*, in Hockey and Ide (1994), 32-45.

[ROB 00] Robinson, P, Project Edition of *Collate*
URL : http://www.cta.dmu.ac.uk/projects/collate/intro.html

[TEX_A] *Archives Tex pour la Devenagari*:
ftp://ftp.tex.ac.uk/tex-archive/language/devanagari/velthuis/doc/