[1]Elaboration d'édition critique de textes sanskrits

Marc Csernel[1], François Patte[2]

[1] Université Paris-Dauphine-1 place du M[al] Delattre de Tassigny-75016 Paris Cedex 16 & Inria Rocquencourt, projet AXIS, Domaine de Voluceau 78 153 le Chesnay Cedex

[2] Université Paris Descartes, 12, rue de l'Ecole de Médecine - Paris 6e

Abstract

To exhibit the differences between different versions of the same text is quite an easy operation for a computer program if it concerns texts written in any modern European language such as English or French, where words are well separated by spaces. On the contrary, if it concerns a language such as Sanskrit, where traditionally no spaces occur between words, and some very long compound words exist, it becomes a difficult problem, especially if there is no valid and reliable computer oriented lexicon available. In this article we intend to show how we proceed to determine, in terms of words, the differences that can exist between the different versions of the same Sanskrit text, with the aim to elaborate a critical edition. Exhibiting these differences, it becomes possible to elaborate a distance between the different versions of the text. Once this step is achieved, the study and determination of relations, which can exist between the versions, becomes possible through the use of phylogenic trees.

Keywords : critical edition, Sanskrit, intertextual distances, longest common subsequence.

Résumé

Etablir les différences entre deux textes, en termes de mots, peut apparaître comme un traitement informatique évident dans une langue européenne moderne comme le français où les mots sont identifiés par des espaces séparateurs. Dans une langue comme le Sanskrit, où, traditionnellement, les frontières entre les mots ne sont pas déterminées par des séparateurs, et où les mots composés peuvent être extrêmement longs, le problème prend une toute autre dimension, surtout si on ne dispose pas d'un lexique informatique fiable. Dans cet article nous indiquons comment procéder pour déterminer, en terme de mots, les différences entre des versions différentes d'un même texte Sanskrit, afin d'établir l'édition critique de ce texte. A partir de la mise en valeur de ces différences, il devient possible de générer des distances intertextuelles. Après cette étape, la détermination et l'étude de filiations entre les différentes versions du texte devient possible via la construction d'arbres phylogénétiques.

1. Introduction

Etablir la différence qui existe entre deux textes en termes de mots peut apparaître évident dans une langue comme le français où les mots sont séparés par des blancs. La différence entre « *Le chat est mort* » et « *Le petit chat est mort* » (Poquelin J.-B. 2003) apparaît triviale non seulement pour un lecteur attentif de l'école des femmes, mais aussi pour un programme informatique relativement simple, qui compare les textes caractère par caractère, et qui « sait » qu'une séquence de caractères commençant et se terminant par un espace délimite un

[1] Cet article a reçu le soutien de l'A.C.I. CNRS « histoire des savoirs », ainsi que du projet Européen IT&C-Asia 2004/091-775.

mot. Ceci est exact non seulement en français mais dans toutes les langues européennes modernes.

Lorsque ces différences apparaissent entre les versions d'un même texte, soit que ce texte comporte différentes éditions, soit qu'il ait été recopié à la main avec toutes les erreurs due à ce genre de procédé, et que ce texte a quelque importance littéraire, religieuse, scientifique..., se pose alors, pour le philologue, la question de faire une édition critique.

Une édition critique est une édition où l'on fait apparaître toutes les différences relevées entre les différentes versions d'un texte en terme d'ajouts, de suppressions, de changements de mots (ou d'ensemble de mots, tels des paragraphes...). Pour la plupart des langues européennes, il existe depuis longtemps des outils informatiques d'aide à la création de telles éditions, par exemple Collate de Peter Robinson (1994). Ils ont permis de créer des éditions critiques par d'autres moyens que les méthodes traditionnelles voir par exemple le travail de O'Hara & Robinson (1993) sur les contes de Canterbury. Ils permettent d'aider le philologue dans l'accomplissement de cette tâche qui peut devenir longue et fastidieuse lorsque le nombre de versions différentes du texte à prendre en compte est important ; dans ce cas, l'édition devient un ouvrage où le texte de l'édition prend moins de place que les notes qui l'accompagnent. Depuis peu, des versions interactives d'édition critiques existent, telle celle proposée dans le projet Digital Nestle-Aland de l'université de Münster, qui permettait de voir une édition interactive de l'évangile selon St Jean dans sa version grecque (St Jean (2003-2007)) sur son site internet jusqu'à une date récente.

Notre but est d'aider les philologues dans l'établissement d'éditions critiques de textes dont la langue est le sanskrit ou toute autre langue possédant des caractéristiques similaires. En sanskrit, où traditionnellement la frontière entre les mots n'apparaît pas, où les mots composés peuvent comporter des centaines de caractères (donc sans séparateurs), le problème prend une toute autre dimension que dans nos langues européennes. Si on ne dispose pas d'un lexique informatique fiable et adapté au sujet le problème paraît inextricable. Par exemple le lexique sanskrit créé par G. Huet (Huet 2004, 2006), qui a demandé des années d'efforts, est parfaitement adapté au vocabulaire des enfants qui apprennent à lire, mais pas aux textes scientifiques ou philosophiques dont nous pensons nous occuper. Un programme qui n'a pas de lexique ni de séparateur pour déterminer les frontières des mots se heurte à deux problèmes majeurs :

- La notion de mot devient floue, il faut essayer de se baser sur d'autres notions telles les phonèmes pour essayer de les déterminer. Le programme ne peut proposer qu'un choix parmi un ensemble de solutions possibles.
- Le nombre de solutions à examiner croît de manière exponentielle en fonction de la longueur des chaînes de caractères, les temps de traitement suivent naturellement. On se trouve avec des programmes qui, du fait de leurs temps d'exécution trop longs, n'arrivent plus à apporter une aide efficace aux philologues.

Nous allons essayer de résoudre ces problèmes grâce à l'emploi d'un texte lemmatisé.

2. L'utilisation d'un texte lemmatisé

Nous avons décidé de bâtir nos éditions critiques à partir d'un texte sous deux formes différentes : une forme lemmatisée où tous les mots apparaissent de manière séparée, et une forme non lemmatisé. Nous appelons le texte lemmatisé, *padapāṭha,* du nom d'une forme de récitation des textes sanskrits faite en détachant toutes les syllabes, et le texte non lemmatisé *samhitapāṭha*, en raison d'une forme de récitation du sanskrit, faite sans détacher les syllabes,

de manière coulée. Le *padapāṭha* (lemmatisé) sera transformé dans le texte de l'édition (non lemmatisé) : le *samhitapāṭha*.

Dans le *samhitapāṭha* les frontières entre les mots n'apparaissent plus nécessairement et de longues séquences de lettres sans séparateur peuvent exister. Lors de la transformation du *padapāṭha* en *samhitapāṭha,* l'emplacement des frontières entre les mots est soigneusement conservé afin de pouvoir indiquer par la suite, dans la comparaison du *samhitapāṭha* avec un autre texte, quels sont les mots modifiés.

Enfin les textes des manuscrits, que nous appelons *mātṛkāpāṭha*, sont comparés l'un après l'autre avec le *samhitapāṭha* (texte de l'édition). Les différences sont exprimées en termes de mots manquants, ajoutés ou changés... Les *mātṛkāpāṭha* sont saisis sous forme d'un fichier « texte », dans lequel apparaissent aussi toutes les remarques et annotations du collationneur.

Lors de la comparaison, le *padapāṭha* constitue un lexique implicite, par rapport auquel les comparaisons de tous les manuscrits vont s'effectuer. C'est seulement grâce à la mise en mémoire des séparations trouvées dans le *padapāṭha* que le programme peut déterminer quels sont les mots qui ont été modifiés, omis, ou ajoutés. Notons que les mots ajoutés n'étant pas lemmatisés, le programme se contente d'effectuer des hypothèses, c'est le travail de l'éditeur de fournir le texte ajouté sous forme de suite de mots et non pas sous la forme d'une simple chaîne de caractères.

Les signes qui servent à la lemmatisation sont le +, le _ et le ^. Du fait de l'application des *sandhi* (cf. §3) la disparition de ces signes lors de la transformation du *padapāṭha* en *samhitapāṭha* entraîne des modifications dont on peut voir un exemple dans la figure 1.

Le + indique une séparation entres les mots déclinés:
padapāṭha: siddhi**s+**v.rtti**s+**iya.m
samhitapāṭha: siddhi**r**v.rtti**r**iya.m
Le _ indique une séparation entre les différents composants d'un mot composé.
Le ^ indique la présence d'un préfixe. Un même mot peut avoir plusieurs préfixes différents.
padapāṭha: v**i^**u**d^**panna_ruupa_siddhi**s+**।
samhitapāṭha: v**y**u**t**pannaruupasiddhi**.h**

Fig 1 : Exemple de transformation de padapāṭha en samhitapāṭha.

Nous avons fait apparaître en gras les lettres transformées par le *sandhi*. Nous voyons dans le premier exemple qu'un **s** se transforme en **r**, dans le troisième qu'un **i** se transforme en **y**.

3. Le problème des *sandhi*

On pourrait penser que l'essentiel des problèmes qui président à l'élaboration de programmes informatiques produisant une édition critique de textes sanskrits est résolu par l'introduction d'un texte lemmatisé, mais il n'en est rien. D'une part le sanskrit s'écrit à l'aide d'un alphabet de 48 lettres translitérées suivant un codage mis au point par Franz Velthuis (Velthuis 1991) ; une lettre de sanskrit translittérée correspond donc à 1, 2 ou 3 caractères latins ; ce qui entraîne nécessairement un prétraitement pour ne pas comparer les lettres latines de la translittération mais effectivement les lettres de l'alphabet sanskrit. D'autre part le sanskrit offre, pour le profane en la matière, une particularité surprenante : celle des transformations morpho-phonétiques appelées *sandhi.*

Ce qu'on appelle *sandhi* – du Sanskrit « liaison » – est un ensemble de règles phonétiques, qui s'appliquent à la jonction de deux morphèmes à l'intérieur d'un mot, ou à la jonction de

deux mots à l'intérieur d'une phrase. Ces règles sont parfaitement codifiées dans la grammaire de *Pāṇini*, mais elles peuvent devenir compliquées du point de vue informatique.

Par exemple la syllabe finale « *as* » est souvent changée en « *o* » si elle suivie d'une sonore, ainsi le mot *tapas* (pénitence) devient *tapo* quand il est suivi par le mot *dhana* (richesse) pour bâtir le mot composé *tapodhana* (celui qui est riche de ses pénitences), alors qu'il reste sous la forme *tapas* quand il est suivi du suffixe « *vin* » : *tapasvin* (un ascète). Le logiciel que nous proposons n'est pas capable pour l'instant de résoudre des problèmes de ce type, qui s'apparentent plus à une exception qu'à ce que nous appelons une règle de grammaire (bien que ces *sandhi* soient mentionnés dans les règles de *Pāṇini)*.

D'autres *sandhi* moins compliqués posent un autre type de problème, la disparition d'un mot complet : la séquence *: iti+i_kareṇa* du *padapāṭha* devient dans le *samhitapāṭha : itīkareṇa*, le mot « i » du composé *i_kareṇa* a disparu !

4. Les données et les résultats attendus

Notre travail s'effectue sur un ensemble important de données, constitué d'une centaine de manuscrits d'un célèbre traité de grammaire nommé *Kāśhikāvṛitti* ou « Glose de Bénarès », qui constitue l'un des plus anciens commentaires de la grammaire de *Pāṇini*. La grammaire de *Pāṇini* forme la base de l'étude linguistique du sanskrit, et constitue le premier exemple connu de grammaire générative. Ces manuscrits sont écrits en utilisant différentes écritures, parmi lesquelles la *devanagārī* prend la place du lion ; ils sont également écrits avec les écritures du sud de l'inde, comme l'écriture *tĕlugu* ou tamoule. Un équivalent européen de l'utilisation de ces différentes écritures serait de voir des textes latins écrits avec les alphabets latin, grec ou cyrillique.

Chaque *mātṛkāpāṭha* est saisi par un groupe de deux lettrés, l'un lisant le texte du manuscrit, l'autre modifiant le *samhitapāṭha* (texte de l'édition) et ajoutant des commandes de collation pour obtenir le *mātṛkāpāṭha* correspondant au manuscrit. La saisie s'effectue chapitre par chapitre et, pour chaque chapitre, un responsable est chargé de bâtir le *padapāṭha* qui, une fois transformé en *samhitapāṭha*, deviendra le texte de l'édition.

Les résultats fournis par notre logiciel sont de deux ordres, d'une part les informations suffisantes pour pouvoir bâtir une édition critique, d'autre part les informations nécessaires pour pouvoir établir des distances entre les textes des manuscrits, et par là même, pouvoir effectuer des analyses de données sur le corpus. En constatant qu'une édition critique peut se décrire en termes de mots ajoutés, supprimés ou modifiés, on remarque que cette description est très proche de celle de la distance d'édition existant entre deux chaînes de caractères. Les résultats de notre logiciel de comparaison, tels qu'ils apparaissent dans l'exemple 1 (pour un mot donné) sont facilement transformables en une distance d'édition (en terme de mots), ils pourront être utilisés pour toutes opérations de classification : partitionnement, construction de hiérarchie, d'arbre phylogénétique, tels ceux décrits dans Barthélemy et Guénoche (1991).

5. La comparaison des textes

C'est une comparaison complexe, car, du fait de la présence des *sandhi*, les textes à comparer ne sont pas homogènes entre eux. La comparaison se fera donc en deux parties :

- La première partie consiste en un prétraitement lexical en deux étapes, qui purge les *mātṛkāpāṭha* des commandes de collation et transforme le *padapāṭha* en un *samhitapāṭha* virtuel qui sera ensuite comparé à un *mātṛkāpāṭha*.

```
Word 14 'lak.sa.nena' is:
       Missing in manuscript st1, tri55
       Substituted with 'ak.sa.nena' in manuscript io2
       Followed by Added word(s) 'ca' in manuscripts ba2, G1
       Followed by Added words 'laaghava.m bhavatiiti' in manuscripts bh1, hss,
       LD0, my5, tri49
       Followed by Added words 'laaghava.m bhavati' in manuscripts cm6, my6mal,
       my6nan, tri21, tri32
```

Exemple 1 : résultats de la comparaison du samhitapāṭha, avec plusieurs manuscrits.

- La deuxième partie consiste, à l'aide de l'algorithme L.C.S. « Longest Common Subsequence » (Hunt J.W. & Szymanski T.G. (1977)), à aligner les séquences du *samhitapāṭha* virtuel (le texte de l'édition) et du *mātṛkāpāṭha* (le texte du manuscrit) afin de déterminer quelle sont les frontières des mots dans le *mātṛkāpāṭha*. Nous pouvons noter que l'algorithme L.C.S constitue aussi les bases du célèbre programme de comparaisons de textes `diff`, dont la version actuelle est basée sur un papier de Myers (1986). Notons néanmoins que `diff` utilise la L.C.S. dans un but tout à fait différent du nôtre.

5.1. Le prétraitement lexical

Nous allons décrire les traitements lexicaux que nous devons effectuer pour rendre le *padapāṭha* et un *mātṛkāpāṭha* comparables. Les traitements s'effectuent à l'aide d'analyseurs lexicaux de type `LEX` qui permettent de reconnaître des expressions régulières et de leur faire subir un traitement approprié. Nous distinguons deux sortes de *sandhi*, ceux qui génèrent des espaces et dont le résultat est susceptible d'être réinterprété et les autres.

```
as+/{SON}              Add("o"); AddSpace();
as+/{VOWEL_A}          Add("a"); AddSpace();
as+a                   Add("o.a");
aas+/{VOWEL}           Add("aa");AddSpace();
as+/(k|p|s|.s|"s)      Add("a.h"); AddSpace();
ai/+{VOWEL}            Add("aa");AddSpace();
ai(_|^)/{VOWEL}        Add("aay");
```

Exemple 2 : Règles concernant les sandhi « génératifs »

L'exemple 2 contient une partie des règles des *sandhi* : « génératifs » sous forme d'expressions régulières `LEX` dans lesquelles le signe « / » signifie « suivi de » ; les termes en majuscule tels `VOWEL`, se réfèrent à des ensembles de lettres définis ailleurs dans le corps du programme.
La fonction `Add` peut être considérée comme une fonction de substitution et `AddSpace` comme une fonction d'ajout d'espace ; enfin la barre « | » représente une alternative.

```
m/{DENTA}                          Alter = LN; return LMPO;
.m/{LABIA}                         Alter = LM; return LMPO;
(a|aa){LEMM}(a|aa)                 return  LABAR;
(a|aa){LEMM}(o|au)                 return  LAU;
.r/{LEMM}({VOWEL}|{DIPH})          return  LR;
e{LEMM}a                           Next = LAVA: return LE:
```

Exemple 3 : règles concernant les sandhi ordinaires

La première règle signifie que la séquence « as+ » (où + est un des signes de lemmatisation) suivie d'une sonore sera transformé dans la voyelle « o » suivi d'un espace.

Ce traitement est effectué avant l'application des règles générales de constitution des *sandhi* : L'exemple 3 contient quelques exemples des règles qui président à ce traitement. Dans cet exemple `LEMM` signifie un signe de lemmatisation, `DIPH` une diphtongue. Les codes retournés sont des codes internes de lettres correspondant à l'alphabet *devanagārī* dans sa translittération Velthuis : `LMPO` par exemple signifie « Lettre M avec un point » : la lettre **.m**. Le terme `Alter` indique que la lettre affectée pourra être substituée à la lettre retournée, les deux versions étant absolument équivalentes. Le terme `Next` indique que la lettre affectée suivra la lettre retournée, bien qu'elle n'apparaisse pas dans le texte original. La première ligne indique que la lettre **m** (suivant le code Velthuis) lorsqu'elle est suivie d'une dentale devient la lettre **.m**, mais qu'elle peut aussi bien devenir la lettre **n** (grâce au terme `Alter`). La troisième ligne indique qu'un **a** ou un **aa** (a long noté *ā* suivant la translittération traditionnelle) suivi d'un signe de lemmatisation puis d'un autre **a** ou d'un **aa,** devient un seul **aa.** Cette règle est porteuse de futurs problèmes car au moment de reconstituer la limite entre les mots on ne saura plus si l'on avait un **a** suivi d'un **aa** ou un **aa** suivi d'un **a** ou tout autre combinaison de ces deux lettres.

Une fois le prétraitement lexical terminé, les deux textes de la comparaison sont homogènes, les limites entre les mots fournies par le *padapāṭha* sont conservées et on peut commencer les alignements.

5.2. L'utilisation de la L.C.S.

Avant de décrire la manière de procéder, il convient de rappeler la définition d'une distance d'édition entre deux chaînes de caractères, *X* et *Y*. C'est le nombre minimum d'opérations nécessaires pour passer de *X* à *Y*, chaque opération pouvant être la suppression, l'ajout, ou la transformation d'une lettre.

Pour trouver quels sont les mots qui différent entre le texte de l'édition (*samhitapāṭha*) et le texte d'un manuscrit, la technique que nous employons est proche des techniques employées pour comparer les séquences moléculaires en biologie. Elle est basée sur l'algorithme bien connu Longuest Common Subsequence (L.C.S.), qui permet d'obtenir une, ou plusieurs, des plus longues séquences communes entre deux chaînes de caractères *X* et *Y*. Cet algorithme bâtit, en utilisant la technique de l'algorithme de la programmation dynamique, une matrice *T* ou le texte *X* apparaît en ligne, et *Y* en colonne, le *i*-ème et le *j*-ième caractère de chaque chaîne sont notés *X[i]* et *Y[i]* respectivement.

Chaque *T[i,j]* de la matrice contient le nombre de caractères communs entre les *i* premiers caractères de la chaîne *X* et les *j* premiers caractères de la chaîne *Y* ; le coin en bas à droite de la matrice contient la longueur de toutes les L.C.S. possibles entre *X* et *Y*. Cette valeur est parfois notée *lcs*(*X,Y)* et peut être considérée comme le dual de la distance d'édition entre *X* et *Y* calculée sans utiliser d'opération de transformation. Chacun des *T[i,j]* est calculé en utilisant l'algorithme de la programmation dynamique suivant la formule :

$$T[i,j] = \begin{cases} T[i-1,j-1] & \text{si x[i]= y[j]} \\ Max(T[i,j-1],T[i-1,j]) & \text{dans les autres cas} \end{cases}$$

Formule 1

Le lecteur attentif aura observé qu'une édition critique peut être considérée comme une distance d'édition formulée en termes de mots entre plusieurs textes, et que, par un curieux

détour de la pensée, pour obtenir cette distance, nous nous servons d'une distance d'édition formulée en termes de lettres entre les chaînes de caractères des textes à comparer. L'exemple de la figure 2 montre comment nous procédons pour comparer deux séquences sanskrites :

X = `"sriiga.ne"saayanama.h` et Y = `tasmai "srii_gurave namas`

La seconde (*Y)* appartient au *samhitapāṭha,* mais contient les limites produites par le *padapāṭha* sous forme de blanc ou de tiret bas (*underscore*), elle sert de lexique et de guide pour la comparaison. La première (*X)* contient une phrase du *mātṛkāpāṭha* que l'on désire comparer avec le texte de l'édition. Les limites des mots trouvés dans le *padapāṭha* apparaissent sous formes de barres horizontales dans la matrice. Chacune des cases de la matrice contient une valeur calculée suivant la formule 1 qui, de manière plus pragmatique, contient pour la case [*i,j*] le nombre de lettres communes entre les *i* premières lettres de *X* et les *j* première lettres de *Y*. Le coin en bas à droite de la matrice contient le nombre de lettre de la L.C.S de *X* et *Y*. On note parfois cette valeur (unique) *lsc(X,Y) ;* elle vaut 11 dans notre exemple.

			"		i			.		"	a							.
			s	r	i	g	a	n	e	s	a	y	a	n	a	m	a	h
t	|	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
a	|	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1
s	|	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1
m	|	0	0	0	0	0	1	1	1	1	1	1	1	1	1	2	2	2
ai	|	0	0	0	0	0	1	1	1	1	1	1	1	1	1	2	2	2
"s	|	0	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2
r	|	0	1	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
ii	|	0	1	2	3	3	3	3	3	3	3	3	3	3	3	3	3	3
g	|	0	1	2	3	4	4	4	4	4	4	4	4	4	4	4	4	4
u	|	0	1	2	3	4	4	4	4	4	4	4	4	4	4	4	4	4
r	|	0	1	2	3	4	4	4	4	4	4	4	4	4	4	4	4	4
a	|	0	1	2	3	4	5	5	5	5	5	5	5	5	5	5	5	5
v	|	0	1	2	3	4	5	5	5	5	5	5	5	5	5	5	5	5
e	|	0	1	2	3	4	5	5	6	6	6	6	6	6	6	6	6	6
n	|	0	1	2	3	4	5	5	6	6	6	6	6	7	7	7	7	7
a	|	0	1	2	3	4	5	5	6	6	6	6	7	7	8	8	8	8
m	|	0	1	2	3	4	5	5	6	6	6	6	7	7	8	9	9	9
a	|	0	1	2	3	4	5	5	6	6	6	6	7	7	8	9	10	10
.h	|	0	1	2	3	4	5	5	6	6	6	6	7	7	8	9	10	11

Fig 2 : Comparaison de deux séquences de caractères

Le rectangle gris foncé en haut à gauche, correspond au mot `tasmai` qui manque dans *Y*, les 2 rectangles gris moyen correspondent aux mots présents à la fois dans *X* et *Y* : `"srii` et `nama.h` (`nama.h` et `namas` sont équivalent du fait d'un *sandhi*), enfin le rectangle gris clair du centre correspond au remplacement de `gurave` par `ga.ne"saaya`. Parmi tous les alignements possibles proposés par l'algorithme L.C.S. nous choisissons celui qui reflète les informations fournies par les rectangles que nous venons d'indiquer. Un tel alignement est représenté figure 3, c'est un des alignements possibles que nous recherchons.

Le nouveau rectangle situé à droite du rectangle gris clair, contenant `"saaya` dans sa partie basse, et qui n'apparaissait pas dans la figure 2, indique une différence possible d'interprétation. En effet en l'absence de connaissance du sanskrit le rectangle gris clair de la figure 2 peut fournir trois interprétations différentes et même plus si on le souhaite :

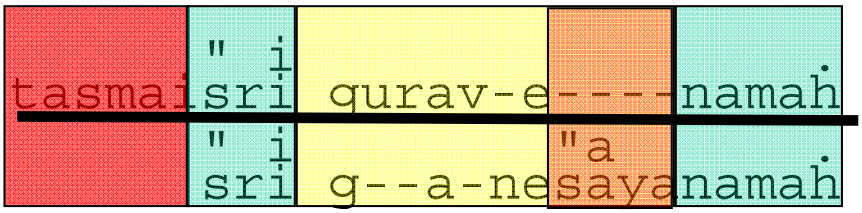

Fig. 3 : Un des alignements possible

- `gurave` est remplacé par `gane"saaya`
- `gurave` est remplacé par `gane` et `"saaya` a été rajouté
- `gurave` est omis, `gane"saaya` est ajouté

Le logiciel choisi la 3e solution (voir l'exemple 4) qui n'est pas forcément la plus heureuse.

```
Word 1 'tasmai' is :
     - Missing
Word 2 '"srii' is :
     -Followed by Added word    'ga.ne"saaya'
Word 3 'gurave' is :
     - Missing
```

Exemple 4 : les résultats fournis par le logiciel pour l'exemple de la figure 2

Comment procéder pour obtenir de tels résultats ? Il existe deux approches possibles :

- Obtenir le bon alignement en trouvant des règles de navigation fines à l'intérieur de la matrice, en fonction du contexte.
- Obtenir tous les alignements possibles et choisir « le meilleur ».

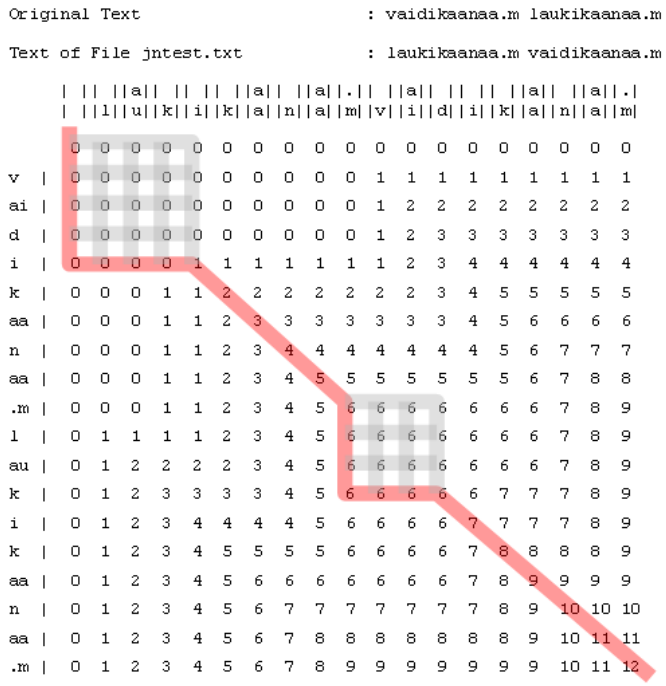

Figure 4 : Illustration du nombre des alignements possibles

Avant de discuter plus avant, regardons la figure 4 qui donne une idée du nombre de chemins différents possibles permettant d'obtenir des alignements. Ces chemins figurent en gris clair

dans la matrice, en gris foncé figure le chemin choisi. On voit que le nombre de chemins possibles croit très rapidement lorsque se présentent des rectangles correspondant à une différence totale (pas de lettre correspondante) entre deux sous chaînes. Si deux carrés ayant dix cases de côté sont présents dans la matrice le nombre d'alignements possibles est de l'ordre de $39*10^9$.

Les problèmes inhérents aux choix possibles sont :

- Pour la première méthode : nous n'avions pas, au début de ce projet, de critères permettant d'évaluer de manière pertinente les résultats obtenus.
- Pour la deuxième méthode : le nombre d'alignements possibles croît d'une manière incroyablement rapide.

Nous avons donc choisis donc de combiner les deux approches :

- Formuler quelques règles de navigation simples permettant de limiter de manière efficace le nombre de solutions.
- Parmi ces solutions possibles, en calculer un certain nombre (quelques milliers) et les évaluer suivant un critère indépendant des règles de navigation choisies. Nous avons choisi un critère lié à la compacité des alignements obtenus.

```
Word 14 'lak.sa.nena' is:
   Missing in manuscript st1, tri55
   Substituted with 'ak.sa.nena' in manuscript io2
   Followed by Added word(s) 'ca' in manuscripts ba2, G1
   Followed by Added words 'laaghava.m bhavatiiti' in  manuscripts
           bh1, hss, LD0, my5, tri49
```

Exemple 5 : exemple de résultats obtenus

Cette approche s'est révélée satisfaisante (l'exemple 5 donne un exemple des résultats que le logiciel peut obtenir) mais elle n'est pas suffisante. Une fois les alignements trouvés, un travail supplémentaire est encore nécessaire afin de présenter les résultats sous une forme qui soit lisible et compréhensible pour un sanskritiste (ou toute autre personne intéressée). Un bon exemple est donné par les alignements des figures 5 et 5 bis. Figurent en grisé les alignements affichables tels quel, après la comparaison, en gras ceux qui nécessitent une réorganisation avant d'être affichés.

```
|     a|" i|                    |    .|
|ta-smi|sri|gu-----r--------ave|namah|
|      |   |                    |     |
|  a  a|" i|  a i a "   ata     |    .|
|t-asmi|sri|g-udiparsvanahaya--|namah|
```

Figure 5 : Les alignements trouvés

```
|     a|" i|      |                |    .|
|ta-smi|sri|gurave|----------------|namah|
|      |   |      |                |     |
|  a  a|" i|      | a i a "   ata  |    .|
|t-asmi|sri|------|gudiparsvanahaya|namah|
```

Figure 5bis : Les alignements souhaités

La figure 5 montre les alignements obtenus après comparaison, la figure 5 bis les alignements souhaités. Nous voyons que le principe de compacité des alignements, a présidé la aussi à la transformation. Le résultat obtenu en suivant la figure 5bis est :

- `tasmai` est changé en `taasmai`
- `gurave` est manquant

- `gaudiipaar"svanaathaaya` est ajouté.

5.3. Les différences avec diff.

Le lecteur peut se demander pourquoi nous n'avons pas choisi d'utiliser l'algorithme `diff` ou d'autres algorithmes connus de comparaison de chaînes de caractères pour obtenir nos résultats. Ces algorithmes, maintenant utilisés non seulement pour comparer les chaînes de caractères mais aussi pour les comparaisons de séquences moléculaires telles celles du génome, vivent, grâce à ces dernières applications, une nouvelle jeunesse. L'exemple 6 montre la différence entre les résultats obtenus par `diff` et les nôtres.

```
1c1                          1d                Word 1 'tasmai' is :
<"sriigane"saayanama.h     <tasmai               - Missing
---                        4c3,5             Word 2 '"srii' is :
>tasmai"sriiguravenama.h   <gurave             -Followed by Added word
                           -----               'ga.ne"saaya'
                           > gane            Word 3 'gurave' is :
                           > "                  - Missing
                           > saaya

diff sans espace    diff avec espace     nos résultats sans espaces
```

Exemple 6 : Les différences de nos résultats avec ceux de `diff`

Cette différence n'est naturellement pas due à la supériorité intrinsèque de nos méthodes, mais à la différence des buts poursuivis. Le but de `diff` est de montrer le plus rapidement possible, sur de très grands ensembles de données, les différences qui peuvent exister entre les chaînes. Il ne s'agit pas de faire « dans la finesse » mais de ne rien manquer. Dans les implémentations habituellement proposées, rien ne permet d'effectuer les réglages qui nous intéressent, et qui concernent principalement la longueur des chaînes de caractères à distinguer dans les résultats.

Au contraire, notre but est, en prenant le temps et tout l'espace mémoire nécessaire, d'arriver à mettre en regard la séquence du texte fournie par le *samhitapāṭha*, muni des séparations fournies par le *padapāṭha*, pour découper le *mātṛkāpāṭha* en une séquence de mots. Ces mots sont ensuite comparés avec ceux du *samhitapāṭha* et le résultat est affiché moyennant quelques adaptations pour le rendre plus lisible. Ayant pu effectuer les réglages qui nous étaient nécessaires, il n'est pas étonnant, bien que nos méthodes dérivent du même algorithme que `diff`, que nos résultats soient nettement supérieurs.

Notre comparaison est pourtant entachée d'un péché originel, nous n'avons pas songé au début de notre conception, que si un mot disparaissait, les *sandhi* qu'il contribuait à former devaient disparaître eux aussi. La correction de cette erreur risque d'entraîner une refonte sérieuse du logiciel.

6. Conclusion

La comparaison des textes sanskrits pour en faire une édition critique est un effort qui se rapproche curieusement de la comparaison de séquences moléculaires telles celles du génome. L'édition critique elle-même devient l'expression d'une distance entre chaque manuscrit et le texte de l'édition formulée, en termes de mots ajoutés transformés ou supprimés. Nous avons donc l'équivalent d'une distance d'édition classique mais en termes de mots, cela permet de réaliser toutes tous les processus de classification possibles entre manuscrits.

Enfin, pour pouvoir effectuer un affichage moderne et satisfaisant de l'édition critique il a été nécessaire d'élaborer des méthodes interactives d'affichage. A ce sujet nous avons pu remarquer que l'utilisation convenable des caractères de l'alphabet *devanagārī* n'est pas toujours très aisée.

Références

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

Buneman P. (1971). *Filiation of Manuscripts Mathematics in Archeological en Historical Sciences.* Edinburgh University Press.

Charras C. & Lecroq T. *Sequence Comparison*.

http://www-igm.univ-mlv.fr/~lecroq/seqcomp/seqcomp.ps

COLING Workshop on Electronic Dictionaries, Geneva, 2004, pp. 8-14.

Crochemore M., Hancart C., Lecroq T. (2001). *Algorithmique du texte,* chapitre 7, pages 223-264. Vuibert, Paris.

Hagel S. Classical Text Editor. http://www.oeaw.ac.at/kvk/cte/

Huet G. (2006) : Héritage du Sanskrit : Dictionnaire Français-Sanskrit.

http://sanskrit.inria.fr/Dico.pdf.

Huet, G. (2004) : Design of a Lexical Database for Sanskrit.

Hunt J.W. & Szymanski T.G. (1977). A fast algorithm for computing longest common subsequence *CACM* 20 : 5.

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

Myers E.W. (1986) : An O(ND) Difference Algorithm and its Variations. *Algorithmica* Vol. 1 No. 2, 1986, p. 251.

O'Hara, R. J. and Robinson P. (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.)

Poquelin J. B. dit Molière. (2003). *L'Ecole des femmes,* Acte II, Sc. 5. Librio, Paris.

Robinson P. (1994). Collate: A Program for Interactive Collation of Large Textual Traditions. In Hockey and Ide, 32-45.

Renou L. (1996). *Grammaire sanskrite : phonétique, composition, dérivation, le nom, le verbe, la phrase*. Maisonneuve réimpression, Paris.

St Jean (2003-2007) Digital Nestle-Aland.

Westfälische Wilhelms-Universität MünsterSchlossplatz 2 48149 Münster

http://nestlealand.uni-muenster.de/AnaServer?NAtranscripts+0+start.anv

Velthuis F. (1991). *Devanagari for TeX, version 1.2 (User Manual)*. University of Groningen, May 1991.