Tri informatique pour le lingala et le hausa dans le projet BTML

Le projet BTML de l'Office québécois de la langue française vise à construire une banque de terminologie multilingue capable de soutenir sur le plan technologique les caractéristiques spécifiques de diverses langues. Le présent article présente un compte rendu des expériences du traitement de langues africaines dans le cadre de ce projet, notamment pour ce qui concerne le tri multilingue à l'aide de la méthode de classement à multiples clés recommandée par la norme ISO/CEI-14651. Des exemples concrets de mise en œuvre de cette méthode pour le lingala et le haussa montrent ses avantages et, aussi, sa limite.

Termes-clés :
banque de terminologie multilingue ; traitement informatique de langues africaines ; classement à multiples clés ; ISO/CEI-14651 ; Unicode.

Introduction

Le projet BTML (banque de terminologie multilingue) de l'Office québécois de la langue française vise à concevoir et à créer une banque de terminologie qui facilite le partenariat international pour la production et le partage de terminologies multilingues et qui profite pleinement des nouvelles technologies de l'information et des communications telles que l'Internet, le multimédia, le traitement informatique des langues naturelles et le support des langues nationales. Selon les principes du véritable multilinguisme, une telle banque doit être capable non seulement de supporter les différentes langues mais en plus de respecter leurs caractéristiques propres et de permettre une grande souplesse aux partenaires. Pour mettre en œuvre ces principes, il a fallu résoudre des problèmes théoriques, tels que la définition d'une banque de terminologie multilingue, la relation entre les langues dans une telle banque, et des problèmes techniques, notamment ceux concernant le codage et les polices de caractères, le tri selon les particularités des langues, l'aide à la saisie des caractères spéciaux, la neutralisation de signes et de caractères non essentiels pour faciliter la recherche, la localisation des interfaces, etc., sans oublier la prise en compte des exigences politico-législatives propres à l'environnement où le système sera utilisé (par exemple, chaque administrateur du système doit pouvoir définir facilement une langue permanente tout en assurant une égalité d'accès et de fonctions à toutes les langues traitées). Dans les lignes qui suivent, nous nous en tiendrons à une discussion sur quelques aspects des tris informatiques en différentes langues et nous présenterons notamment un bilan de nos tests pour le lingala et le hausa basés sur la norme ISO/CEI-14651 (2001).

Problématique du tri et principes de notre solution

Quand on parle du tri, le plus souvent, on pense à une simple mise en ordre des lettres d'un alphabet, comme s'il existait un ordre préétabli et transcendant. Mais, en réalité, le tri est beaucoup plus complexe que l'expression « de A à Z » ne le suggère. S'agissant du tri en informatique, la question se complique davantage en raison de la diversité des finalités, des destinataires et des environnements du traitement où s'inscrit l'opération du tri. Ainsi, le tri peut viser simplement à classer des éléments dans un ordre *quelconque*, à l'aide d'un algorithme déterministe pour que des opérations informatiques comme la comparaison et la recherche puissent s'effectuer de façon fiable, c'est-à-dire de façon à aboutir toujours aux mêmes résultats étant donné les mêmes conditions d'exécution. Ce genre de tri est destiné à la machine et l'aspect linguistique et culturel n'y compte guère. Tous ceux qui ont utilisé MS-DOS se rappellent ce programme de tri, *SORT*, qui classe les lettres et les autres signes purement et simplement selon la valeur numérique de leur code. Pour les 26 lettres de base de l'alphabet latin, les minuscules sont donc rangées après « Z » majuscule ; quant aux caractères accentués, quand ils sont disponibles, ils sont tous placés après les lettres de base, sans aucune logique apparente quant à leur ordre. Ce tri fonctionne sans problème pour la machine, d'autant plus que la distribution des codes est plus ou moins faite pour optimiser le traitement informatique. Par exemple, pour les 26 lettres de base, un écart de 32 (en décimal) sépare les lettres en majuscule et leur correspondant en minuscule, ce qui permet de les convertir d'une casse à l'autre à l'aide d'une opération facile et rapide de OR ou XOR. Toutefois, si l'on doit utiliser le résultat de ce genre de tri pour afficher les données destinées à l'humain, l'absence totale du

respect des habitudes linguistiques et culturelles devient alors inconvenant, frustrant, voire révoltant. En effet, l'ordre dans lequel les locuteurs d'une communauté linguistique ont l'habitude de classer et de chercher les éléments de leur langue fait partie de leur manière d'utiliser cette langue, au point qu'une déviation peut être perçue comme une anomalie ou même un barbarisme, et peut devenir un obstacle qui empêche un usage efficace, cohérent et correct des résultats du traitement informatique. Ainsi, si quelqu'un cherche un mot dans une longue liste chevauchant plusieurs pages et s'attend à trouver classés de façon consécutive tous les mots commencés par « a », toutes casses confondues, avec ou sans signe diacritique, le résultat d'un tri selon la valeur décimale des codes risque de l'induire en erreur en le poussant à abandonner prématurément la recherche dès qu'il voit une entrée commencée par « B ».

Or, effectuer les tris informatiques en tenant compte de la diversité linguistique et culturelle des destinataires humains n'est pas une tâche facile, du fait que le tri informatique se doit d'être absolument prévisible, déterministe et fiable alors que les habitudes humaines dans la réalité des langues naturelles ne sont pas toujours de cette rigueur. Pour l'alphabet latin, si la convention est plus ou moins bien établie quant à l'ordre du classement des caractères de base de A à Z, beaucoup de langues peuvent avoir des caractères particuliers qui ne se trouvent pas dans cet alphabet de base. Le français a ses caractères accentués et ses bigammes soudés, sans oublier d'autres signes de toutes sortes qui peuvent être utilisés dans son écriture. Même s'il existe des conventions non écrites suivies de façon plus ou moins rigoureuse par les grands dictionnaires, il n'est pas certain que les règles du tri soient connues et partagées par tous. Si l'on demande à des locuteurs francophones comment il faut classer des mots qui ne se distinguent que par des signes diacritiques à différentes positions dans les mots, ou comment il faut classer les mots précédés par des chiffres ou par des lettres grecques comme « α » et « β », il y a fort à parier qu'on obtiendra des réponses différentes. Le chinois pose un problème d'une autre nature, parce qu'il utilise un grand nombre de caractères idéographiques au lieu d'un alphabet ayant un nombre bien restreint de lettres. Force est de combiner plusieurs niveaux de critères. Il existe plusieurs méthodes différentes à cet effet : on peut commencer par un classement selon les radicaux pour ensuite ordonner les caractères d'un même radical selon le nombre de traits composants, puis la prononciation, etc. ; on peut aussi ordonner les caractères d'abord selon les codes représentant les graphèmes des quatre coins du caractère pour ensuite considérer d'autres critères ; les dictionnaires modernes utilisent le plus souvent un classement qui considère d'abord la prononciation, puis le ton, ensuite le nombre de traits et, s'il y a lieu, le radical. Toutefois, quelle que soit la méthode combinatoire utilisée, il peut toujours y avoir une zone floue où des éléments ne se distinguent plus selon les critères appliqués. L'arbitraire et la variation trouvent alors leur place. Dans le cas d'une langue comme le coréen, qui a pourtant un alphabet restreint appelé ***hangŭl***, les conventions du tri peuvent varier au niveau des lettres consonantiques ou vocaliques dites doubles, par exemple « ㄲ » (kk) par rapport à « ㄱ » (k), ou « ㅖ » (ye) par rapport à « ㅔ » (e). Certains préfèrent les classer comme des éléments de tri indépendants l'un de l'autre, d'autres préfèrent les regrouper. On pourrait donc voir, à titre d'illustration schématique, des séquences de tri du genre « 게 계 께 꼐 » si on préfère séparer les consonnes simples et doubles mais regrouper les voyelles simples et doubles (ce qui correspond à la pratique la plus courante), ou « 게 께 계 꼐 » si on préfère l'inverse.

Adapter les tris informatiques aux destinataires humains, c'est aussi tenir compte de la diversité des besoins selon les contextes d'utilisation. On peut s'en tenir à des éléments de tri qui se distinguent par leur forme graphique seulement, ou opérer sur des éléments de tri qui sont des regroupements logiques basés sur des critères sémantiques, phonétiques, thématiques, typographiques, etc. Par exemple, selon les cas, on peut préférer classer « ph » comme « f » en se basant sur un critère phonétique, ou comme deux caractères séparés selon un critère purement orthographique. Pour le français, on peut vouloir classer ensemble les paires de mots comme « clé » et « clef », à titre de variantes d'un même élément, si on opère sur un niveau lexical, tout comme on peut regrouper « e, é, è, ê... » en les considérant comme des variantes d'un même caractère de base. Selon que l'on tienne compte ou non des conventions typographiques, on peut traiter « oe » et « ae » comme des séquences de caractères séparés, quand c'est écrit ainsi, ou appliquer systématiquement une analyse pour déterminer s'il s'agit de bigammes soudées « œ » et « æ » (cet exemple, présenté dans ce sens, concerne notamment la comparaison et la recherche, car au niveau du tri proprement dit, la question se pose le plus souvent dans l'autre sens : classer les

bigammes soudées comme deux caractères séparés ou comme un seul caractère à part).

Enfin, l'adaptation des tris informatiques aux destinataires humains doit faire face à un obstacle de taille: la diversité parfois déroutante des «cultures informatiques». En effet, la grande variété des systèmes d'exploitation, les multiples jeux de caractères, les langages de programmation, les familles de SGBD (système de gestion de bases de données), les langages de script pour la publication sur le web (ASP, PHP, JavaScript, VBscript), les logiciels ou les progiciels d'application spécifique... ce sont autant de «fiefs» où il peut y avoir une «culture» spécifique ayant une influence favorable ou défavorable sur les tris informatiques. Ainsi, le monde humain, qui a créé le monde informatique, cherche maintenant à adapter celui-ci à ses besoins, sous peine de se faire aliéner de sa particularité linguistique et culturelle. À en croire Marc Wilhelm Küster (2000), la difficulté pour classer correctement en informatique les «ch», «ll» et «ñ» de l'espagnol en tant qu'éléments à part après les «c», «l» et «n» respectivement aurait même amené à réviser la norme traditionnelle du classement en espagnol pour l'adapter à la tyrannie informatique.

Ce tour d'horizon de la problématique du tri est de nature à nous indiquer quelques principes pragmatiques pour guider notre recherche de solution en matière de tri informatique adaptable aux besoins de diverses langues dans notre projet BTML.

– Le tri étant variable selon son utilité, son destinataire et son contexte d'application, il convient de bien délimiter les besoins du tri dans différentes parties de notre système afin de diminuer l'ampleur des difficultés et de conserver les caractéristiques particulières propres à chaque langue là où l'effet vaut la peine de notre effort. Plus précisément, il convient de concentrer notre effort sur l'adaptation du tri à différentes langues là où les destinataires humains doivent parcourir la liste des éléments linguistiques dont on suppose normalement qu'ils sont classés selon l'habitude de la langue en question. En revanche, dans des parties du système où la comparaison et la recherche à l'aide d'un tri informatique concernent seulement l'ordinateur, il est alors superflu de vouloir imposer la culture humaine. Par exemple, à un moment donné il nous a fallu exporter une longue liste de mots d'exclusion à partir du SGBD vers un programme en VBScript du côté du serveur web. Des comparaisons et des recherches très rapides doivent être effectuées par l'ordinateur entre les mots clés entrés par les utilisateurs et les mots de cette liste. Pour cette raison, la liste doit être triée. Au niveau du SGBD, le programme du tri est adapté à l'aspect linguistique et culturel de chaque langue; mais au niveau du VBScript, le tri par défaut suit des règles qui ne respectent pas les caractéristiques des langues humaines. Dans un cas pareil, vouloir imposer le respect des cultures humaines à VBScript semblerait une lutte don-quichottesque.

– Pour une banque de terminologie multilingue, qui relève d'applications spécifiques, il convient de déterminer une procédure commune à toutes les langues et facile à mettre en œuvre pour adapter les règles du tri à divers besoins. Par contre, lorsqu'il existe plusieurs conventions de tri pour une même langue, il n'est pas nécessaire pour nous de les intégrer toutes dans le système. En fait, le tri n'est pas une fin en soi, mais un moyen pour faciliter la comparaison et la recherche. Or, les moyens de recherche exacte ou floue à l'aide de l'ordinateur deviennent de nos jours tellement évolués que si, en appliquant une méthode du tri parmi d'autres selon les particularités d'une langue, on ne peut toujours pas permettre aux utilisateurs de consulter facilement la liste des résultats, ce sera alors au niveau conceptuel du processus d'interrogation et de recherche qu'il y a lieu de mettre un peu plus d'effort.

Solution de tri basée sur la norme ISO/CEI-14651

Suivant l'analyse qui précède, nous avons décidé d'adopter les directives et le modèle de classement de la norme ISO/CEI-14651 comme cadre méthodologique pour adapter et mettre en œuvre les règles du tri selon les langues.

Pour résumer, cette norme remonte aux travaux d'Alain LaBonté[1], informaticien-conseil au Secrétariat du Conseil du trésor du Gouvernement du Québec. Au cours des

1 L'auteur de cet article voudrait exprimer son remerciement à Alain LaBonté pour avoir accepté de relire et de corriger les paragraphes concernant ses travaux et la norme ISO/CEI-14651 dont il a été le rédacteur. Les lecteurs intéressés par la question du tri informatique en contexte multilingue sont invités à lire le texte original de la norme et les publications d'Alain pour obtenir des informations techniques plus détaillées et plus précises.

années 80, dans sa recherche d'une méthode de tri informatique absolument prévisible quant aux résultats et respectueuse de l'aspect linguistique et culturel du français, il détermina qu'il était nécessaire d'effectuer une décomposition à plusieurs niveaux de l'information pour en rendre le classement prévisible de manière absolue et, aussi, permettre d'établir un tri dont la précision pouvait varier selon les besoins. Pour intéresser les producteurs de logiciels, il essaya de démontrer qu'une telle méthode pourrait s'appliquer à plusieurs autres langues et il fit un parallèle avec le traitement à plusieurs niveaux utilisé dans les dictionnaires chinois. Il fit aussi un parallèle entre la structure informatique des nombres à virgule flottante, où l'on peut, en effectuant une comparaison uniquement sur l'exposant exprimé, classer les nombres selon leur ordre de grandeur, la mantisse venant au besoin ajouter plus ou moins de finesse au résultat des comparaisons...

Ainsi il a mis au point une méthode de classement qui permet de trier le français avec haute précision et grande souplesse. Sa méthode s'appuie sur l'utilisation de clés de comparaison à quatre niveaux définis selon leur degré de précision, à savoir, le niveaux des caractères de base, celui des signes diacritiques, celui de la casse et enfin celui des autres signes distinctifs. Sauf le premier niveau, les autres sont optionnels et leurs clés peuvent faire l'objet de réduction selon certaines techniques, ce qui réduit la longueur des clés et le poids de stockage.

Ses travaux ont ensuite conduit à une norme canadienne du classement, CAN-CSA Z243.4.1, qui permet de traiter adéquatement à la fois l'anglais, le français, le portugais, l'allemand, l'italien et le néerlandais. Sous son instigation, le projet ISO/CEI-14651 fut créé en 1992 et il en a été le rédacteur. Après de longues années de travail ardu, souvent contre vents et marées, il a enfin réussi à faire émerger un consensus dans les pays membres de l'ISO et chez tous les géants incontournables de l'industrie informatique, autour de la nécessité d'une norme internationale du classement et autour de la pertinence de sa méthode de base. Celle-ci, après des années de révision, s'est améliorée et s'est élargie pour devenir une méthode universelle d'adaptation des règles de comparaison et de classement des données informatiques. La norme a finalement été adoptée en 2001. Selon une étude, cette norme est la douzième norme internationale de l'informatique la plus citée dans le monde (la première étant la norme du jeu universel de caractères codés sur plusieurs octets (JUC), soit la norme ISO/CEI-10646, avec laquelle le développement d'Unicode est harmonisé).

Inutile de préciser qu'il ne s'agit point pour cette norme de fournir des règles du tri prêtes à appliquer à toutes les langues, mais bien d'une méthode d'adaptation uniforme, cohérente et facile d'emploi des règles du tri pour toutes les langues. La norme comporte en annexe quatre tables de symboles de tri (déjà classés) relevant des quatre niveaux sus-mentionnés, plus une table commune de classement, basé sur ces symboles, de tous les caractères du JUC. Ces symboles et leur classement tiennent compte des particularités des différentes écritures utilisées par les langues du monde et conviennent déjà à beaucoup de contextes courants pour des destinataires humains. Pour toute langue qui a des besoins spécifiques en tri, la norme fournit les directives pour élaborer un fichier delta qui répertorie les symboles, éléments et classement faisant l'objet d'adaptation à un niveau ou à un autre. L'adaptation, qui porte souvent sur un nombre limité de symboles et d'éléments pour une langue donnée (excepté le cas du chinois), peut naturellement bénéficier du modèle des tables offertes par la norme, de sorte que, lorsqu'on utilise Unicode pour construire une application à portée multilingue, il est possible de réaliser des tris informatiques qui soient à la fois adaptés à une langue en particulier et compatibles avec d'autres langues en général.

Dès lors, il nous a paru essentiel de trouver une façon d'appliquer la méthode d'adaptation de cette norme à notre système de façon à ce que par la suite, quand le besoin se ferait sentir, l'adaptation particulière pour une langue puisse s'effectuer isolément et facilement. Sur ce point, le travail se trouve grandement facilité par l'architecture globale de notre système dans laquelle chaque langue est traitée dans une base à part, tandis qu'une base centrale coordonne ce qui touche à toutes les bases. Ainsi, nous pouvons facilement instaurer soit des règles communes à plusieurs bases à l'aide de la base centrale, ou des règles spécifiques à une langue dans la base de celle-ci. L'application d'une méthode universelle et cohérente basée sur une norme internationale conduit dès le début à créer des programmes et fonctions partageables par toutes les langues concernées même si elles nécessitent des règles de tri différentes.

Ce qui suit a été relativement simple. D'une part, nous avons analysé les données et repéré les catégories de données qui peuvent être affichées aux utilisateurs sous forme de liste triée. Pour toutes ces catégories de données, nous avons créé des champs en double avec une longueur trois fois plus grande que la longueur initiale (selon nos tests, les clés ne dépassent pas cette longueur) pour y stocker les clés de tri. D'autre part, avec plus ou moins d'adaptation selon les cas, nous avons créé les tables des symboles de tri pour les quatre niveaux, fixé une méthode pour attribuer les poids aux symboles de tri de façon à ce que les valeurs des poids de chaque niveau ne chevauchent pas celles des autres niveaux, créé les tables (adaptées) du classement où chaque élément de tri possède quatre clés générées selon les poids des symboles de tri utilisés. Parallèlement à ces tables de classement, nous avons créé une fonction qui peut être appelée avec comme paramètres une chaîne de caractères et une indication du niveau et qui retourne la clé complète du niveau indiqué pour la chaîne en question. Si nous voulons un tri à deux niveaux, il suffit de concaténer les clés des deux niveaux ainsi obtenues, et ainsi de suite. Le cas échéant, nous pouvons même changer l'ordre des niveaux ou sauter certains niveaux intermédiaires pour des besoins spécifiques. S'il faut générer les clés d'un certain niveau avec un balayage à rebours, comme c'est le cas pour le français au deuxième niveau selon la norme canadienne du classement, il suffit d'inverser la chaîne de caractères en intrant ou la clé générée en extrant.

Une fois que nous avons établi la procédure d'adaptation des règles du tri selon la norme ISO/CEI-14651 et créé les programmes et fonctions nécessaires à la fabrication des clés de tri pour chacun des quatre niveaux, nous avons mené des tests sur le lingala et le hausa en collaboration avec les membres du LLACAN (Edema Atibakwa, Christian Chanard, Marcel Diki-Kidiri), pour fins de démonstration dans le cadre d'un projet du Rifal. Tous les tests ont été réalisés dans un environnement de Microsoft, avec Windows 2000 serveur, SQL 2000 et IIS 5. Ce choix s'explique par le simple fait que cette plate-forme supporte Unicode au niveau natif, ce qui simplifie beaucoup notre travail pour développer une application à portée multilingue (ce n'est pas pour rien que la norme JUC est la plus citée, comme nous l'avons mentionné plus haut).

Tests du tri pour le lingala et le hausa

Les informations linguistiques de base concernant les règles du tri en lingala et en hausa nous ont été fournies par Christian et Edema. En guise de résumé, voici la liste des caractères de ces deux langues, avec toutes les combinaisons possibles des signes diacritiques, que nous devions parvenir à classer dans l'ordre indiqué:

Pour le lingala

a à á â ǎ b d e è é ê ě ɛ ɛ̀ ɛ́ ɛ̂ ɛ̌ f i ì í î ǐ k l m mb n nd ŋ ng ɲ ny nz o ò ó ô ǒ ɔ ɔ̀ ɔ́ ɔ̂ ɔ̌ p s t ts u ù

Pour le hausa

a à á ā ă â b ɓ c d ɗ e è ê ē ĕ ê f fy g gw gy h i ì î ī ĭ ĩ j k kw ky ƙ ƙw ƙy l m n o ò ô o ŏ ô p r r̃ s sh t ts u ù û ū ŭ ũ w y 'y z

Ces caractères peuvent varier en casse, bien entendu, le majuscule devant précéder le minuscule selon l'information fournie.

Il existe plusieurs écoles de pensée en matière de conventions régissant le classement des caractères dans ces deux langues. Les uns préfèrent classer les caractères selon leur graphie, les autres selon leur valeur phonologique. Pour fins de test, nous avons choisi de suivre l'école qui classe les caractères selon leur valeur phonologique, car ce classement est plus exigent pour le traitement informatique et convient mieux à nos tests. Notons que pour l'objectif de nos tests et dans l'état actuel du développement des normes de base concernant l'orthographe et les règles du tri de ces langues africaines, la question de savoir dans quelle mesure ces informations ont été validées par un consensus plus ou moins large n'est pas pertinente pour nous. Nous prenons pour acquis que lorsque vient le moment pour nous d'intégrer officiellement les règles du tri pour ces langues, la question de la validité des informations linguistiques de base aura été réglée en amont.

Ainsi, pour le lingala, les groupes de caractères « mb », « nd », « ny », « nz » et « ts », qui correspondent à des unités phonologiques à part, doivent être traités comme des éléments de tri simples et, dans un classement, doivent suivre respectivement « m* », « n* », « nd* », « ny* » et « t* ». Autrement dit, par exemple, « mba » doit être classé après « mza », ce que, pour le moment, aucun logiciel commercialisé sur le marché ne peut réaliser par défaut,

bien entendu. Il en est de même pour le hausa, dont les groupes de caractères « fy », « gw », « gy », « kw », « ky », « ƙw », « sh », « ts » et « ʼy » doivent être traités comme des éléments de tri à part et classés dans l'ordre indiqué par la liste montrée plus haut. Nous allons appeler ces groupes de caractères « bigammes » dans le reste du texte.

Mais les particularités des tris informatiques pour ces deux langues ne se limitent pas aux groupes de caractères qui doivent être traités à la manière de bigammes soudées et qui, naturellement, n'existent pas encore comme tels dans le répertoire d'Unicode. En effet, les nombreux signes diacritiques nécessitent aussi un traitement particulier. Dans certains cas, il s'agit de caractères composés à l'aide de signes combinatoires prévus à cet effet dans Unicode ; dans d'autres, on voit des combinaisons de deux signes diacritiques sur un caractère de base.

Bigammes ou caractères composés, leur point commun pour nous, c'est que c'est à nous qu'il incombe de voir à ce qu'ils soient traités comme des éléments de tri soudés. À cet effet, dans un premier temps, il a fallu établir un fichier delta pour documenter les adaptations à effectuer à partir des tables modèles fournies par la norme.

En premier lieu, il a fallu définir autant de nouveaux éléments de tri qu'il y a de bigammes et de caractères composés à classer qui n'existent pas déjà dans le répertoire d'Unicode. Un élément de tri peut être simple ou complexe, c'est-à-dire composés de plusieurs caractères simples. Les caractères « Ǎ » et « ǎ » du hausa, par exemple, suivant les directives de la norme, peuvent être définis comme étant la combinaison des caractères de base « Ā » et « ā » avec le signe diacritique grave « ̀ ». On peut les noter comme suite dans le fichier delta :

```
% Lettre majuscule hausa a avec macron et grave
collating-element <U0100_0300> from "<U0100><U0300>"
% Lettre minuscule hausa a avec macron et grave
collating-element <U0101_0300> from "<U0101><U0300>"
```

Ensuite, il faut adapter les symboles de tri du premier niveau. Un symbole de tri peut être associé à plusieurs éléments de tri. Par exemple, le symbole de tri <S0061> est associé à tous les éléments de tri qui comportent le caractère de base « a », peu importe s'ils portent ou non des signes diacritiques et peu importe s'ils sont au minuscule ou au majuscule. C'est de cette façon qu'on peut obtenir un classement qui est indifférent aux signes diacritiques et aux casses. En l'occurrence, pour le lingala et le hausa, deux approches sont possibles selon qu'on veut ou non pouvoir par la suite faire varier le degré de précision en jouant avec les symboles de tri du deuxième niveau qui concernent normalement les signes diacritiques. D'abord, on peut définir de nouveaux symboles de tri complexes pour les bigammes et les caractères composés, et on leur attribue des poids. Ainsi traités, ces bigammes et ces caractères composés propres à ces deux langues seront comme des caractères de base simples au point de vue du classement. Pour l'exemple de « A avec macron et grave » en hausa, suivant cette approche, on peut alors définir un nouveau symbole de tri minuscule qui va être utilisé pour les deux casses de ce caractère composé :

```
% Lettre minuscule hausa a avec macron et grave
collating-symbol <S0101_0300>
```

En revanche, suivant une deuxième approche, on peut ne pas définir de nouveaux symboles de tri au premier niveau mais, plutôt, on y associe des symboles de tri existants qui correspondent aux caractères de base qu'on peut extraire des caractères composés à traiter. Toujours pour l'exemple de « A avec macron et grave », en hausa, on peut ainsi utiliser <S0061> (« a ») comme symbole de tri au premier niveau. Lors de la définition des clés complètes du classement pour ces caractères composés, on définit au deuxième niveau une clé qui est composée de deux symboles de tri correspondant aux deux signes diacritiques superposés :

```
% Réordonner après lettre majuscule latine a macron
% Lettre a macron et grave
<U0101_0300> <S0061>;"<BASE><MACRO><GRAVE>";
      "<MIN><MIN><MIN>";<U0101_0300>
% Lettre A macron et grave
<U0100_0300> <S0061>;"<BASE><MACRO><GRAVE>";
      "<CAP><MIN><MIN>";<U0100_0300>
```

Cette seconde approche est préférable à notre sens, car elle permet de tirer profit d'une méthode de tri à plusieurs niveaux. Par exemple, si on utilise seulement les clés du premier niveau, on réalisera un tri indifférent aux signes diacritiques ; ajoutant les clés du deuxième niveau, et le tri devient alors sensible aux signes diacritiques. Pour nos tests sur le lingala et le hausa, par commodité, nous avons considéré les seconds caractères des bigammes au même titre que des signes diacritiques des caractères composés au deuxième niveau. Cette seconde approche peut donc nous donner le même type de souplesse pour ces bigammes que

pour les caractères accentués, à l'aide des clés du deuxième niveau.

Quant aux clés des deux derniers niveaux, celles du troisième niveau permettent de faire varier la précision du tri en rapport avec les casses, tandis que celles du quatrième niveau, elles, ont une utilité particulière: elles permettent de réaliser un tri informatique de façon absolument déterministe et prévisible peu importe comment on aura défini les clés aux trois premiers niveaux, car elles s'appuient sur des valeurs uniques correspondant aux positions respectives des éléments de tri dans la table du classement.

Pour les symboles de tri utilisés aux deuxième et troisième niveaux, ainsi que les tables de classement de ces symboles, nous avons simplement modifié l'ordre du classement de certains signes diacritiques (placer l'accent aigu après l'accent grave) et des deux casses (mettre la majuscule avant la minuscule), en suivant littéralement les informations fournies.

Dans le cas des bigammes qui doivent se placer en bloc après certains caractères, il s'agit d'insérer dans la table du classement, à la position voulue, les éléments de tri définis pour ces bigammes. Voici, à titre d'exemple, la structure complète du fichier delta pour le lingala, avec toutes les définitions concernant la bigamme « mb » à divers niveaux:

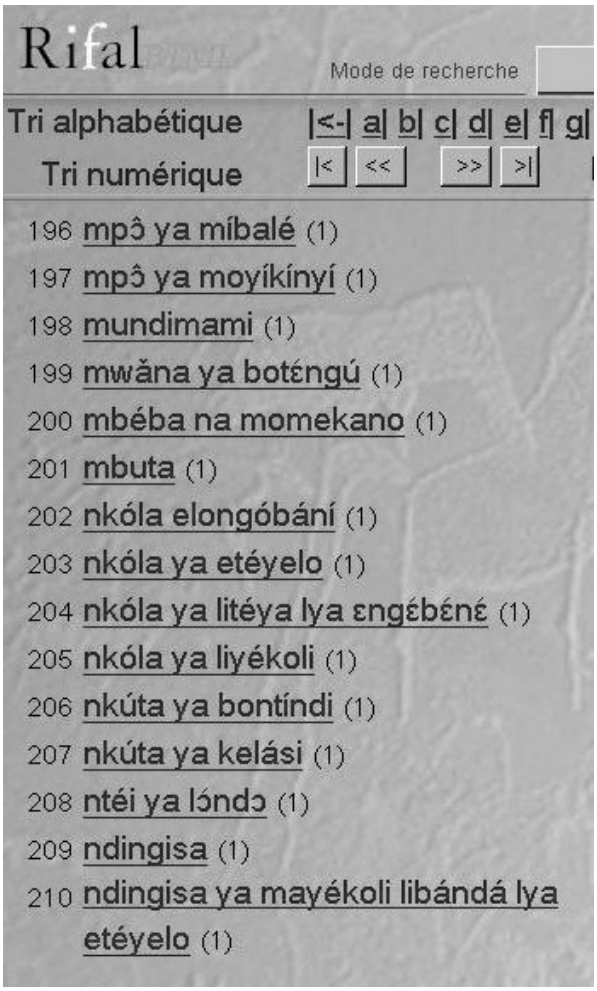
```
% Fichier DELTA pour le lingala
reorder-after <SFFFF>
order_start forward;forward;forward;forward
% copy ISO14651_2000_TABLE1
% Définition de symboles de tri ajoutés pour le lingala
%
% Digraphe minuscule lingala mb
collating-symbol <S006D_0062>
[...]
% Niveau 1: réordonner et/ou ajouter des symboles de tri
reorder-after <S006D>
% Digraphe minuscule lingala mb
<S006D_0062>
[...]
% Niveau 2: inverser l'ordre de <GRAVE> et <AIGUT>
reorder-after <GRAVE>
<AIGUT>
% Niveau 3: inverser l'ordre de <CAP> et <MIN>
reorder-after <CIRCLECAP>
<MIN>
<WIDE>
<COMPAT>
<FONT>
<CIRCLE>
% Définition d'éléments de tri ajoutés pour le lingala
[...]
% Digraphe minuscule lingala mb
collating-element <U006D_0062> from "<U006D><U0062>"
% Digraphe majuscule lingala mb
collating-element <U004D_0062> from "<U004D><U0062>"
[...]
% Ajouter les nouveaux éléments de tri dans la table de tri
[...]
reorder-after <U004D> <S006D>;<BASE>;<CAP>;<U004D>
% Digraphe minuscule lingala mb
<U006D_0062> <S006D_0062>;<BASE>;<MIN>;<U006D_0062>
% Digraphe majuscule lingala mb
<U004D_0062> <S006D_0062>;<BASE>;<CAP>;<U004D_0062>
[...]
reorder-end
% Fin du fichier DELTA pour le lingala
```

Ci-dessous sont présentés des exemples de résultat des tris réalisés après ces adaptations.

Exemple 1: Comparaison des tris avec ou sans adaptation

Adaptation	Classement des éléments	Éléments comptés
Aucune	mbayu	5
	muyu	4
	pingu	5
	pinu	4
Pour le lingala	muyu	4
	mbayu	4
	pinu	4
	pingu	4
Aucune	shuku	5
	sosho	4
	soso	4
	suku	4
Pour le hausa	soso	4
	sosho	4
	suku	4
	shuku	4

Exemple 2 : Tri du lingala (interface BTML)



Exemple 3 : Tri du hausa (interface GDT)

Remarques sur la limite de notre approche

À première vue, la mise en œuvre de la norme ISO/CEI-14651 semble donner des résultats assez satisfaisants. Par exemple, pour le français, notre programme du tri range correctement « école-clinique » entre « école centrale » et « école commerciale », ce que le tri de MS-SQL2000 ne permet de faire en aucun des modes prévus pour le français. Pour le lingala et le hausa, les bigammes sont classées correctement selon les règles indiquées. En plus, elles sont comptées comme des caractères simples par le programme du tri.

Toutefois, nous avons également rencontré des difficultés dues essentiellement à deux types de raisons.

Limite d'une méthode de tri « étrangère » aux systèmes natifs

D'abord, nous avons constaté la limite d'une méthode de tri dont la mise en œuvre demeure « étrangère » aux systèmes natifs (que ce soit le système d'exploitation ou le système de bases de données) sur lesquels l'application est développée et doit s'appuyer pour son fonctionnement. C'est que la réalisation des tris selon les règles spécifiques du classement pour une langue nécessite non seulement la mise en ordre de tous ses éléments, mais aussi la définition de ceux-ci. Les systèmes d'exploitation ou de gestion de bases de données supportent de façon native certaines définitions de certains éléments, permettant ainsi le choix, par exemple, entre plusieurs jeux de caractères, ou entre plusieurs types de classement pour une langue donnée. Lorsqu'un choix est fait, le fonctionnement tout entier de ces systèmes sous-jacents bascule vers les programmes et les ressources adéquats, garantissant ainsi que toutes les opérations du traitement informatique s'effectuent de façon cohérente en fonction du choix. Or, le traitement des langues comme le lingala et le hausa implique la définition *ad hoc* de certains éléments que les systèmes sous-jacents ne reconnaissent que dans la limite des programmes et fonctions que nous avons développés explicitement à cet effet. Hors de cette limite, point de salut.

Dans notre cas, nous pouvons certes afficher les résultats de recherche selon les règles du tri particulières des deux langues africaines testées, mais ce support demeure imparfait, restreint et fragile. Par exemple, quand on voit « sh* » classé correctement après « so » pour le hausa dans l'index des résultats de recherche, il y a en fait déjà une petite anomalie (anodine, faut-il dire, pour notre objectif) : la séquence « sh » n'étant définie comme une bigamme que dans la limite de nos programmes, les deux caractères composants « s » et « h » demeurent toujours deux entités séparées pour les systèmes sous-jacents et sont donc traités comme tels aussitôt qu'on sort de nos programmes et fonctions spécifiques du tri. C'est pour cette raison que lorsqu'on cherche tout ce qui commence par « s », on peut toujours trouver les séquences « sh* », ce qui ne devrait pas être le cas si « sh » était reconnu comme une entité à part entière et différente de « s». Une autre version de cette anomalie peut être observée lorsqu'on cherche tout ce qui se trouve après « so» : cette fois-ci, les séquences « sh* » ne reviennent plus, ce qui représente une inconsistance par rapport à ce qu'on voit avec la recherche par « s». Techniquement, et théoriquement, on sait comment on peut éliminer ce genre d'anomalies : il s'agit, d'une part, de remplacer systématiquement et partout les caractères à comparer et à chercher avec leurs clés de tri dûment générées selon les règles spécifiques et, d'autre part, d'effectuer systématiquement et partout la comparaison et la recherche non pas dans le champ original mais dans le champ correspondant qui contient les clés de tri. Toutefois, dans la pratique, cette solution n'est guère réalisable en dehors des systèmes natifs, parce qu'elle risque d'alourdir excessivement le traitement et est trop coûteuse en termes de ressources informatiques requises. Dans le fonctionnement d'une application, il y a beaucoup d'endroits où les opérations de comparaison et de recherche sont nécessaires ; les systèmes sous-jacents mettent souvent en place des moyens d'optimisation et ce, au cœur même des systèmes, au niveau des codes machine, pour accélérer ces opérations. Appliquer systématiquement et partout une solution *ad hoc* de comparaison et de recherche signifie ni plus ni moins substituer un système étranger à un système natif. Plus une application est complexe, plus cette substitution sera difficile. Dans notre cas, pour le champ des entrées de la table principale d'interrogation, nous avons créé plusieurs champs de recherche où nous stockons les données ayant subi des traitements spécifiques afin d'offrir des méthodes de recherche optionnelles et performantes : neutralisation et substitution de certains

signes et caractères, inversion de la chaîne, nettoyage de balises HTML, etc. Pour garantir une performance satisfaisante lors de ces traitements, nous devons profiter pleinement des moyens d'optimisation offerts par les systèmes natifs. Si nous devions effectuer systématiquement la comparaison et la recherche avec les clés de tri spécifiques de chaque langue, nous devrions alors augmenter énormément le volume de données à traiter et alourdir considérablement chaque opération. Force nous est alors de trouver un équilibre entre des systèmes natifs performants mais incompatibles avec les besoins spécifiques, et une solution idéale pour ceux-ci mais impraticable tant que nous ne pouvons nous passer de ces systèmes natifs.

Complexité de la réalité

La norme ISO/CEI-14651 propose une méthode de classement basée sur une décomposition des données à plusieurs niveaux, allant du moins précis au plus précis. Par défaut, ces niveaux sont définis comme concernant les caractères de base, les signes diacritiques, les casses et tous les autres signes distinctifs. Au cours de nos tests, nous nous sommes rendu compte que dans la vie réelle, les problèmes à résoudre pour un traitement adéquat des données dans le cas d'une banque de terminologie d'envergure alimentée par de nombreuses personnes dans des conditions variées au cours de longues années, posent parfois des « colles » auxquelles ne peut suffire cette décomposition hiérarchique nette et claire en ces quatre niveaux. Plus précisément, il nous a paru que le principe véritablement essentiel et pertinent de la méthode, c'est la décomposition des données selon leur état et selon le degré de précision requis ; en revanche, la nature de ces quatre niveaux et l'ordre dans lesquels ils sont proposés importent parfois peu. L'adaptation de la méthode sur ce point est d'autant plus nécessaire que nous devons surmonter les difficultés dues au manque de support natif à l'égard des règles spécifiques (voir plus haut). Un exemple concret que nous avons vécu fournit une illustration de ce point.

Pour l'anglais et le français, nous avons dans un premier temps utilisé des clés à quatre niveaux, en vue de pouvoir classer adéquatement les termes qui ne diffèrent que par la présence des signes de ponctuation ou des espaces intercalés mais qui doivent tout de même être traités, comptés et affichés séparément en raison de leur association avec les différents domaines et en raison des fiches notionnelles dans lesquelles ils sont inscrits. En plus, il faut harmoniser les comptages effectués au niveau du serveur de données à l'aide des clés de tri et au niveau du programme d'interface qui n'a pas d'accès à ces clés pour des raisons d'optimisation. Citons, par exemple, « plate-forme, plate forme, plateforme », « fire-wall, fire wall, firewall », « ac, a.c. », etc. Par la suite, nous avons remarqué des problèmes liés à la casse : le fait d'inclure la clé du troisième niveau entraîne une distinction systématique dans le traitement des termes qui ne diffèrent que par la casse, partiellement ou entièrement. Cette distinction de casses a une conséquence au niveau du comptage et au niveau du produit de jointure des tables. L'analyse des cas montre qu'il y a trois situations possibles : 1) La différence des casses n'est pas pertinente, étant due à une erreur de saisie ; 2) La différence est pertinente, parce qu'on veut consigner les variations orthographiques, mais les termes renvoient à la même notion (« Cegep » et « CEGEP ») ; 3) La différence est pertinente et les termes renvoient à des notions différentes (« bac » et « BAC »). Dans cette situation, nous ne pouvons pas simplement utiliser une fonction de conversion pour traiter toutes les données en minuscule ou en majuscule ; en outre, nous ne pouvons pas non plus compter sur la possibilité de faire effectuer une correction massive et systématique dans les données à la source. Il ne nous reste plus qu'à essayer d'enlever la clé du troisième niveau tout en conservant celle du quatrième niveau. L'adaptation semble fonctionner, puisque l'indifférence aux casses convient aux situations 1) et 2), et que cette indifférence est compensée par l'unicité des identifiants des fiches notionnelles qui entrent en ligne de compte à un certain point de notre traitement. Suite à cela, nous avons remarqué qu'en fait, il n'y a qu'un tout petit nombre de signes qu'il nous faut prendre en considération au quatrième niveau pour satisfaire à nos besoins. Ce sont l'« espace », le « trait d'union-signe moins », le « point » et la « barre oblique », pour ne citer que ce que nous avons noté jusqu'ici. Inclure le traitement des clés du quatrième niveau juste pour ces signes ne nous paraît pas une solution économique. Nous avons donc décidé de définir des symboles de tri pour ces signes et de les prendre en compte au deuxième niveau. Étant donné que dans la table du classement du quatrième niveau proposée comme modèle

par la norme ISO/CEI-14651, ces signes sont classés avant les signes diacritiques, nous les avons donc insérés immédiatement après le symbole <BASE> au deuxième niveau. Cette adaptation fonctionne pour le moment à notre satisfaction.

En fait, l'adaptabilité de la démarche et des niveaux de la décomposition des données en vue de concevoir les clés adéquates selon les langues et selon les besoins est sous-entendue par la norme ISO/CEI-14651. Universelle, cette norme aurait du mal à se justifier comme telle si elle ne pouvait pas s'adapter et s'appliquer aux besoins d'une langue comme le chinois, par exemple. Or, la décomposition pour les caractères chinois doit se faire sur des niveaux complètement différents, tant par leur nature que par leur nombre, de ceux proposés par la norme pour les autres langues, le chinois n'ayant ni signes diacritiques ni casses. L'éclaircissement de ce point dans notre compréhension de la norme ISO/CEI-14651 en ce qui concerne l'adaptabilité de sa méthode de tri doit, à notre avis, permettre de mieux concevoir et adapter nos solutions selon les besoins et selon les langues.

Conclusion

Les difficultés pour adapter le tri informatique à l'humain s'expliquent par la diversité des besoins, la diversité linguistique et culturelle, la diversité des systèmes informatiques et, bien évidemment, la différence entre le mode de fonctionnement des ordinateurs et celui des êtres humains. Comprendre objectivement tous ces aspects de la problématique du tri constitue déjà un pas en avant dans la recherche d'une solution adéquate et efficace. Tout en maintenant notre principe du respect des caractéristiques propres aux langues traitées, nous pensons également qu'il importe de bien cerner les fonctions et les données pour lesquelles il y a lieu de mettre en œuvre spécifiquement les règles du tri selon les langues, car pour le moment nous ne pouvons le faire qu'en recourant à des méthodes « étrangères » aux systèmes natifs sous-jacents à notre application. Dans cette perspective, la norme ISO/CEI-14651 nous fournit une méthode d'adaptation universelle, cohérente et facile à implanter. Le résultat des tests sur le lingala et sur le hausa nous a confirmé la possibilité de l'utilisation de cette norme dans notre projet BTML et, en même temps, nous a permis de prendre conscience de certaines limites. Sur le plan technique, en dernière analyse, le mécanisme le plus important d'un tri informatique adaptable à la diversité linguistique et culturelle, c'est de décomposer adéquatement les données à classer pour construire des clés en plusieurs niveaux allant du moins précis au plus précis. Le contexte de développement d'une application peut imposer parfois des difficultés imprévisibles ; suivre une approche pragmatique nous semble donc de mise pour pouvoir réaliser des tris informatiques à la fois prévisibles, fiables et adaptables à diverses langues. Il va de soi que toute réalisation du tri informatique doit s'appuyer sur des informations précises en amont quant à la description linguistique des éléments à classer.

Jian Yang,
Chef de projet BTML, Office québécois de la langue française, Québec.
jian@oqlf.gouv.pc.ca

Bibliographie

LaBonté (A.), 1988 :
– *Règles du classement alphabétique en langue française et procédure informatisée pour le tri*, Québec : Publications du Québec.

LaBonté (A.), 1989 :
Technique de réduction – Tris informatiques à quatre clés, Québec : Publications du Québec.
Note : Ces deux publications ont été revues et corrigées par l'auteur en 1998 pour leur version électronique publiée à http://www.tresor.gouv.qc.ca/doc/classm.htm et à http://www.tresor.gouv.qc.ca/doc/techtri.htm respectivement.

ISO/IEC 14651, 2001 :
Technologies de l'information – Classement international et comparaison de chaînes de caractères – Méthode de comparaison de chaînes de caractères et description du modèle commun et adaptable de classement

Küster (M. W.), 2000 :
« Developing European Ordering Rules – The basics of a new sorting standard from A to Ω to å, æ and beyond », dans *MultiLingual computing & Technology*, Volume 11 Issue 5, July/August 2000, p. 33-36.