Journée *Sciences de la communication et informatique multilingue* - 7 octobre 2009

Outillage informatique pour la pratique du plurilinguisme

Jean-François Perrot,

LIP6 - Université Pierre et Marie Curie & ERTIM - INaLCO

Argument

On constate la multiplication des outils informatiques favorisant le multilinguisme,
qui permet de croire au retour de la bonne tradition de l'Europe savante, où chacun pouvait s'exprimer dans sa langue en comptant bien être compris de tous les autres.
Cette floraison s'explique par les progrès de l'ingénierie logicielle, capable de construire des édifices complexes d'une manière modulaire et extensible.
La famille des outils de traduction proposés par Google en est un bon exemple.

Mais la conception serait beaucoup plus difficile et la réalisation infiniment plus fragile en l'absence d'un système de normes et de conventions généralement acceptées.
*La thèse présentée ici est que la généralisation d'*Unicode *est à la base de ce développement.*

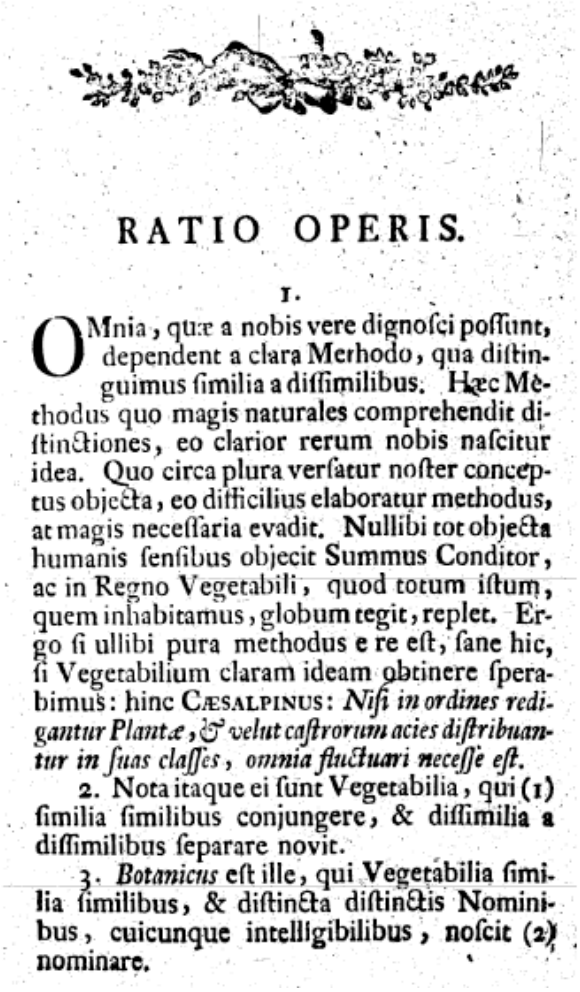
On rappellera donc le problème du codage des écritures et le principe d'Unicode,
et on s'efforcera d'illustrer quelques conséquences de son adoption :

- *Le codage par défaut de la norme XML*
- *L'Internationalisation des URIs et des noms de domaines*
- *La commodité d'outils pour l'accès direct aux textes :*
  exemples en grec (Perseus) et en sanskrit (Inria).

- On adopte ici le point de vue d'une *informatique pour glossophiles* (amateurs le langues, pas nécessairement linguistes),
  en se dégageant de l'obsédante question de la traduction, pour prôner le *retour aux textes originaux*.

- On se limite à l'écrit. Non que l'informatique n'ai rien à proposer en matière d'oralité, mais c'est

une toute autre histoire.
Et par écrit, on entend *texte* au sens des informaticiens (ce sur quoi on peut faire du copier-coller).
Avec tout le respect dû à la BNF, le document reproduit ci-dessous n'est pas un *texte*, mais une *image*...



RATIO OPERIS.

1.

OMnia, quæ a nobis vere dignoſci poſſunt, dependent a clara Methodo, qua diſtinguimus ſimilia a diſſimilibus. Hæc Methodus quo magis naturales comprehendit diſtinctiones, eo clarior rerum nobis naſcitur idea. Quo circa plura verſatur noſter conceptus objecta, eo difficilius elaboratur methodus, at magis neceſſaria evadit. Nullibi tot objecta humanis ſenſibus objecit Summus Conditor, ac in Regno Vegetabili, quod totum iſtum, quem inhabitamus, globum tegit, replet. Ergo ſi ullibi pura methodus e re eſt, ſane hic, ſi Vegetabilium claram ideam obtinere ſperabimus: hinc CÆSALPINUS: *Niſi in ordines rediganturPlantæ, & velut caſtrorum acies diſtribuantur in ſuas claſſes, omnia fluctuari neceſſe eſt.*

2. Nota itaque ei ſunt Vegetabilia, qui (1) ſimilia ſimilibus conjungere, & diſſimilia a diſſimilibus ſeparare novit.

3. *Botanicus* eſt ille, qui Vegetabilia ſimilia ſimilibus, & diſtincta diſtinctis Nominibus, cuicunque intelligibilibus, noſcit (2) nominare.

Le codage des caractères

1. **Information**

C'est ce que contient la mémoire de l'ordinateur : bits (0 et 1) & octets (groupes de 8 bits)
En pratique on ne parle que d'octets :
les octets sont écrits en deux chiffres hexadécimaux (`0 - 9` et `a - f`) chacun représentant un groupe de 4 bits.

Ex. la suite de 64 bits
`0100111101110101011101000110100101101100011011000011000010110011 1`
sera en général montrée sous la forme de 8 octets **`4f 75 74 69 6c 6c 61 67`**

```
0100 1111 - 0111 0101 - 0111 0100 - 0110 1001 - 0110 1100 - 0110 1100 ....

   4f          75          74          69          6c          6c  ....
```

Cette information n'a aucune signification par elle-même !
Tout est affaire d'interprétation...

- Notamment, il est toujours loisible de considérer que ces 64 bits représentent un nombre entier écrit en base 2, à savoir : **`5725610497410883943`**.
  Cette interprétation est la base de tous les procédés actuels de cryptographie à clef publique.

- Une autre interprétation fait correspondre chaque octet avec une lettre de l'alphabet latin.
  On trouve alors : **`Outillag`**

2. **L'équation *octet = caractère***

   Le nombre total d'octets est 256. L'alphabet qu'on apprend à l'école en France compte 26 lettres.
   Il y a donc bien assez d'octets pour coder
   - les 26 lettres,
   - majuscules et minuscules,
   - les 10 chiffres décimaux,
   - toute une batterie de signes de ponctuation,
   - et une poignée de "caractères de contrôle" hérités de la pratique des télégraphistes

   bref tout ce qui est utile à la pratique informatique courante.

   Comme les octets ont sur les machines modernes une sorte de réalité tangible
   (ils sont directement accessibles par les opérations de lecture et d'écriture)
   l'équation *octet = caractère* a pour les programmeurs une valeur de dogme.
   L'évolution qui conduit à Unicode part de cet état de fait.

3. **Le code ASCII (*American Standard Code for Information Interchange*)**

   En fait, pour les besoins courants la moitié suffit :
   les 128 octets du code ASCII réalisent exactement le programme ci-dessus - et rien de plus !
   Aucune lettre accentuée, l'alphabet latin de base et c'est tout.

   Tous les programmes informatiques qui fonctionnent de par le monde,
   et une bonne partie des documents disponibles, notamment des pages Web,
   sont écrits avec ce code.
   Le cas échéant, on recourt à un "surcodage" :
   par exemple, dans une page Web on écrira en ASCII une *entité* `"&eacute;"` et le navigateur
   affichera un e accent aigu.

4. **Les codes sur 8 bits : Latin-1**

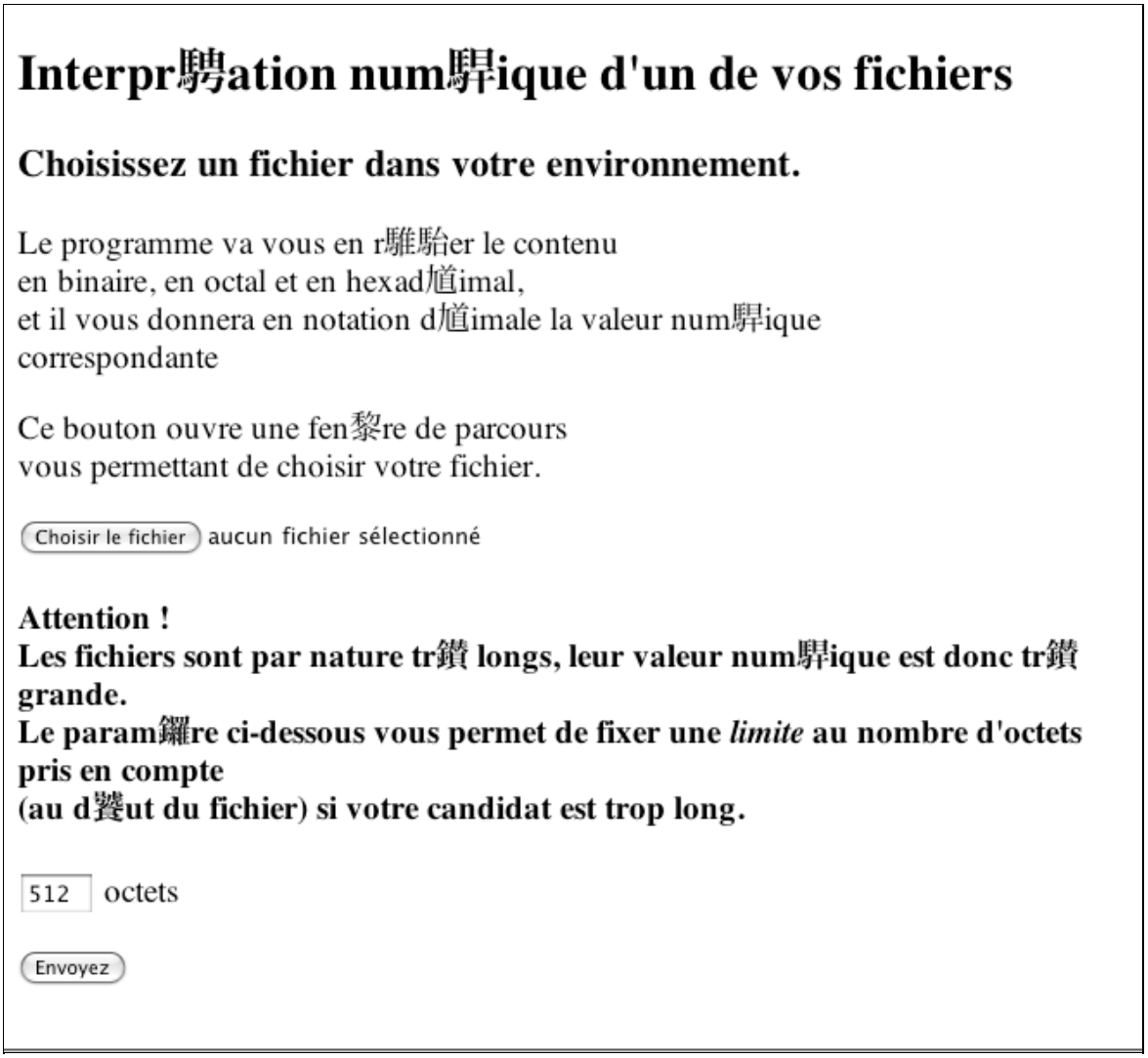
   Très tôt, on a cherché à utiliser les 128 octets restants pour accommoder les lettres diacritées
   (accents, cédilles et autres)
   et les lettres supplémentaires en usage dans les langues à écriture latine autres que l'anglais.
   Et on constate alors que le nombre d'ajouts nécessaires dépasse 128 !
   On arrive donc à proposer plusieurs tables de codage sur 8 bits, chacune s'adressant à un groupe de
   langues : elles ont été normalisées par l'ISO sous le numéro 8859.
   Soulignons que toutes ces tables contiennent sans modification les 128 octets-caractères ASCII.

   La première de ces tables (ISO 8859-1, aussi appelée Latin-1) arrive assez bien à satisfaire les
   besoins de la plupart des langues de l'Europe occidentale, aussi est-elle très utilisée chez nous -
   malgré l'absence du 'œ'.
   La série contient aussi des tables pour le grec moderne, le cyrillique, l'hébreu et l'arabe,
   mais elle ne peut prétendre à couvrir l'ensemble des écritures du monde, en particulier la masse
   des caractères chinois.
   D'autres systèmes ont donc été inventés...

   Cette étape de l'évolution est caractérisée par le foisonnement des systèmes de codage.
   La série ISO-8859 n'est pas adoptée partout où elle pourrait l'être,
   et dans le monde asiatique on trouve aussi plusieurs systèmes de codage des caractères chinois sur
   2 octets.

5. **Illustration de cette diversité**

   Que penser quand une étudiante japonaise se plaint de voir une des pages de votre cours s'afficher
   ainsi ?



Élémentaire, mon cher Watson !
Cette étudiante avait son navigateur réglé sur son codage favori *Shift-JIS*, très utilisé au Japon, alors que votre page était écrite en *Latin-1*, et ne le signalait pas comme elle aurait dû le faire !
En effet :

Shift-JIS (2 octets) Latin-1

- 騁 `e974 = é t`
- 舁 `e972 = é r`
- 雖 `e976 = é v`
- 駘 `e96c = é l`
- 旗 `e963 = é c`
- 黎 `ea74 = ê t`
- 鑽 `e873 = è s`
- 鑼 `e874 = è t`
- 饕 `e962 = è b`

Unicode

Unicode désigne une approche unifiée et universelle au problème du codage des caractères, soutenue par un consortium d'entreprises américaines depuis 1991.

La version actuellement en vigueur est la 5.2.
L'adoption de cette approche, de plus en plus répandue est un facteur essentiel pour le développement de l'outillage informatique pour la pratique du plurilinguisme.

1. **Catalogue : le numéro Unicode**

Unicode recense tous les systèmes d'écriture qui sont attestés sur notre planète, y compris ceux qui ne sont plus en usage.
Ce recensement prend la forme d'une énumération de 1 114 112 caractères potentiels (= 17 × 65.536, soit 17 plans de 2^{16} places chacun), numérotés, en hexadécimal, de `0x000000` à `0x10FFFF`. *Potentiels*, car dans cette liste certains numéros ne sont pas attribués, afin de garder une certaine cohérence dans la numérotation.
En effet, les signes individuels sont regroupés logiquement en blocs (p. ex. arabe, cyrillique, éthiopien), à l'intérieur desquels la numérotation est continue. Entre les blocs se trouvent souvent des plages inutilisées.
En plus de son numéro, chaque signe porte un nom officiel (en majuscules et en anglais) - sauf les caractères chinois.

Exemples :
- Tous les signes recensés dans le code ASCII font partie du catalogue Unicode, avec le même numéro.
- Alpha minuscule 'α' `GREEK SMALL LETTER ALPHA` porte le n° 945 (`x03B1`)
- Le "r voyelle" du sanskrit 'ऋ' `DEVANAGARI LETTER VOCALIC R`, a le n° 2315 (`x090B`)
- Le caractère chinois désignant le soleil '日' a le n° 26085 (`x65E5`)

Comme l'emploi du nom est compliqué et parfois impossible, la désignation officielle d'un caractère unicode est "`U+`" suivi de son numéro *en hexadécimal* à 4 chiffres.
Nos trois derniers exemples ci-dessus sont donc `U+03B1`, `U+090B` et `U+65E5` respectivement, et le A majuscule (ASCII) est `U+0041`.

Le n° Unicode peut s'employer directement pour désigner un caractère dans un contexte ASCII, par exemple dans un programme Java ou JavaScript (sous la forme "`\u`" suivi du numéro en hexadécimal. ainsi pour nos trois exemples `\u03B1`, `\u090B` et `\u65E5`), ou dans une page en HTML, à l'usage d'un navigateur, dans une entité "`&x`*`numéro-hex`*`;`", ainsi `&x03B1;`, `&x090B;` et `&x65E5;`.

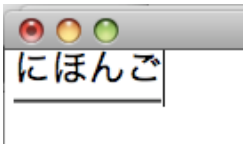
2. **Le caractère comme *objet***

D'un point de vue fonctionnel, le caractère n'est qu'un intermédiaire entre une *saisie* (au clavier, par un dessin, etc) et un *rendu* (affichage à l'écran ou réalisation par l'imprimante).
Sur l'ordinateur

- le rendu est affaire de police (en anglais *font*).
  De la part des informaticiens, la notion de police de caractères a fait l'objet d'une modélisation complexe qui a abouti à la théorie *TrueType* et à sa variante *OpenType* (*TrueType font* = `ttf`), censée assurer la compatibilité entre plates-formes.
  Les ordinateurs modernes sont dotés de collections de polices immenses, si bien qu'en première approximation on peut considérer le problème du rendu comme résolu.

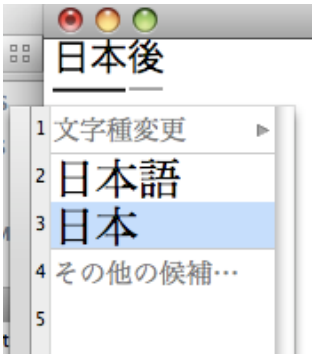
- la saisie est affaire de clavier.
  En effet, l'acquisition par un dessin ne fonctionne que pour certans assistants personnels (*Palm*), et pour les caractères chinois - mais je ne crois pas qu'elle soit d'usage courant.
  Le clavier de l'ordinateur ne fonctionne pas de manière mécanique, il est entièrement programmé.
  On peut donc le reprogrammer *ad libitum* pour associer une combinaison de touches quelconque à un caractère arbitraire.
  Mais l'utilisateur ne change pas facilement ses habitudes digitales !

  De toutes façons, tout clavier reste impuissant devant les caractères chinois.
  On est obligé de recourir à une autre solution, celle de l'intuition par la machine, sur la base d'une frappe approximative, en fonction d'un savoir encyclopédique. Le cas échéant, plusieurs choix peuvent être proposés.
  Exemple en japonais :



1. L'utilisateur a tapé "`nihonngo`", (= *la langue japonaise*) la machine l'a transcrit en *hiragana*.

2. L'utilisateur a tapé *un espace*, la machine propose les kanjis qu'elle estime les plus probables.



3. L'utilisateur a tapé *un second espace*, la machine l'interprète comme un désaveu, elle propose alors un autre choix et elle ouvre la discussion...

Mais Unicode et allé plus loin que cette perspective, en élaborant une théorie du caractère, qui en fait un objet (au sens informatique du terme).
Cet objet possède une batterie de propriétés (p. ex. son sens d'écriture, sa capacité à se lier, etc), support d'algorithmes divers,
ce qui fait du catalogue une vaste base de données consultable par tout un chacun.

3. **Les codages (au pluriel)**

On pourrait se dire que le numéro Unicode suffit pour coder un carctère dans un fichier. Mais ce numéro est un nombre, et comme tel il demande à être représenté : combien d'octets doit-on lui attribuer ? en nombre fixe ou variable ?

- Si on choisit un nombre d'octets fixé, vu que les machines ne connaissent que les puissances de 2, il en faut 4, soit 32 bits.
  Ce codage est connu comme *UTF-32*. Simple mais coûteux en place.

- Si on considère que la très grande majorité des numéros en usage tient sur 2 octets, on adoptera 16 bits comme base.
  C'est ce que font la plupart des langages de programmation, à commencer par Java.
  Mais ce choix (appelé *UTF-16*) conduit à des incompatibilités subtiles (*gros-boutiens* contre *petit-boutiens*), et il a l'inconvénient d'exclure le bon vieil ASCII.
  Aussi est-il à usage interne (dans les buffers des éditeurs), et non pas pour écrire des fichiers destinés à être échangés.

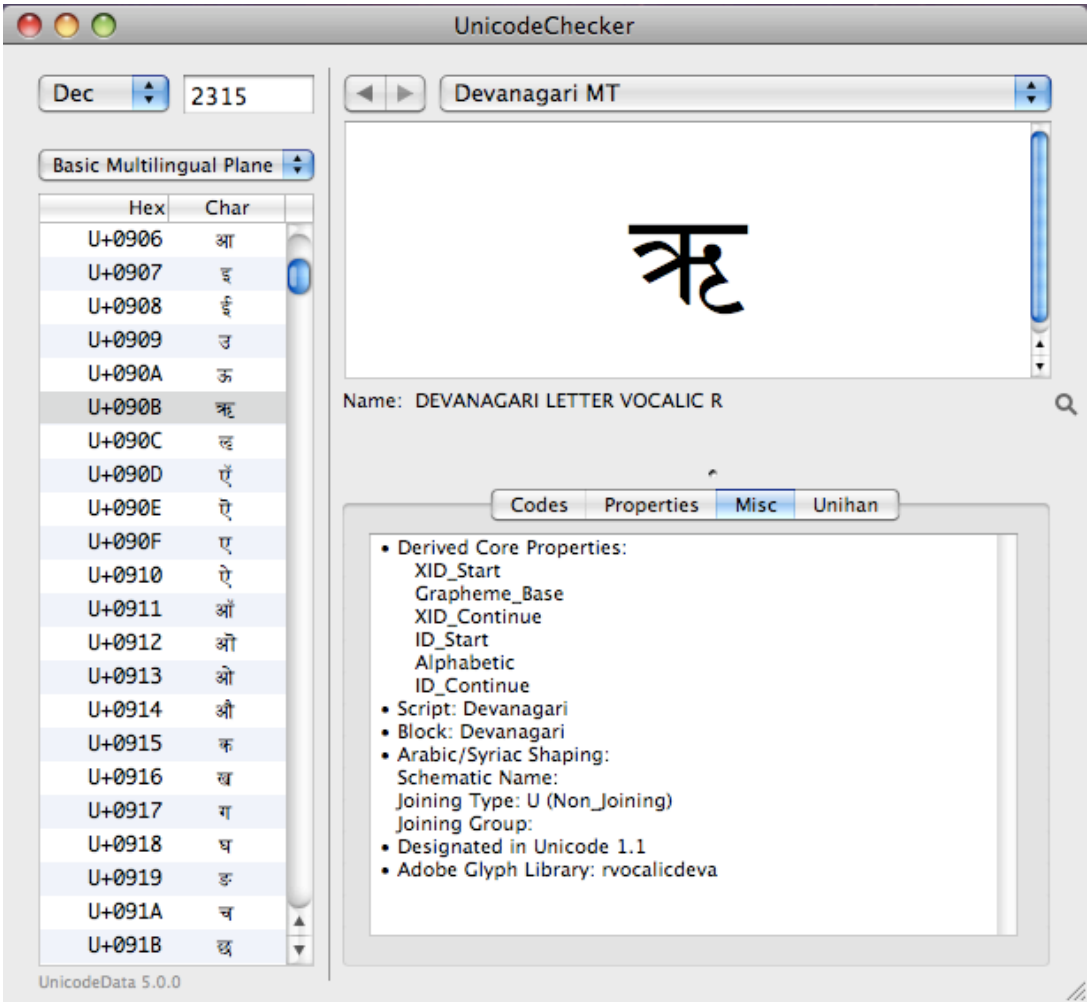
- Le format adopté pour les échanges est *UTF-8* : de 1 à 4 octets, suivant le numéro, avec un algorithme de codage fort simple.
  Nos exemples favoris :
  - 'α' `U+03B1` → `CEB1` (2 octets)
  - '𐤨' `U+090B` → `E0A48B` (3 octets)
  - '日' `U+65E5` → `E697A5` (3 octets)
  - pour dépasser 3 octets il faut aller dans les plans supérieurs d'Unicode, qui sont encore assez peu fréquentés.

  Le format à un seul octet est réservé à l'ASCII, qui se trouve ainsi inclus (un logicien dirait *subsumé*) dans la nouvelle norme.
  Ce dernier point est d'une importance capitale du point de vue des échanges de fichiers, car grâce à lui **tout fichier ASCII est aussi un fichier UTF-8** !

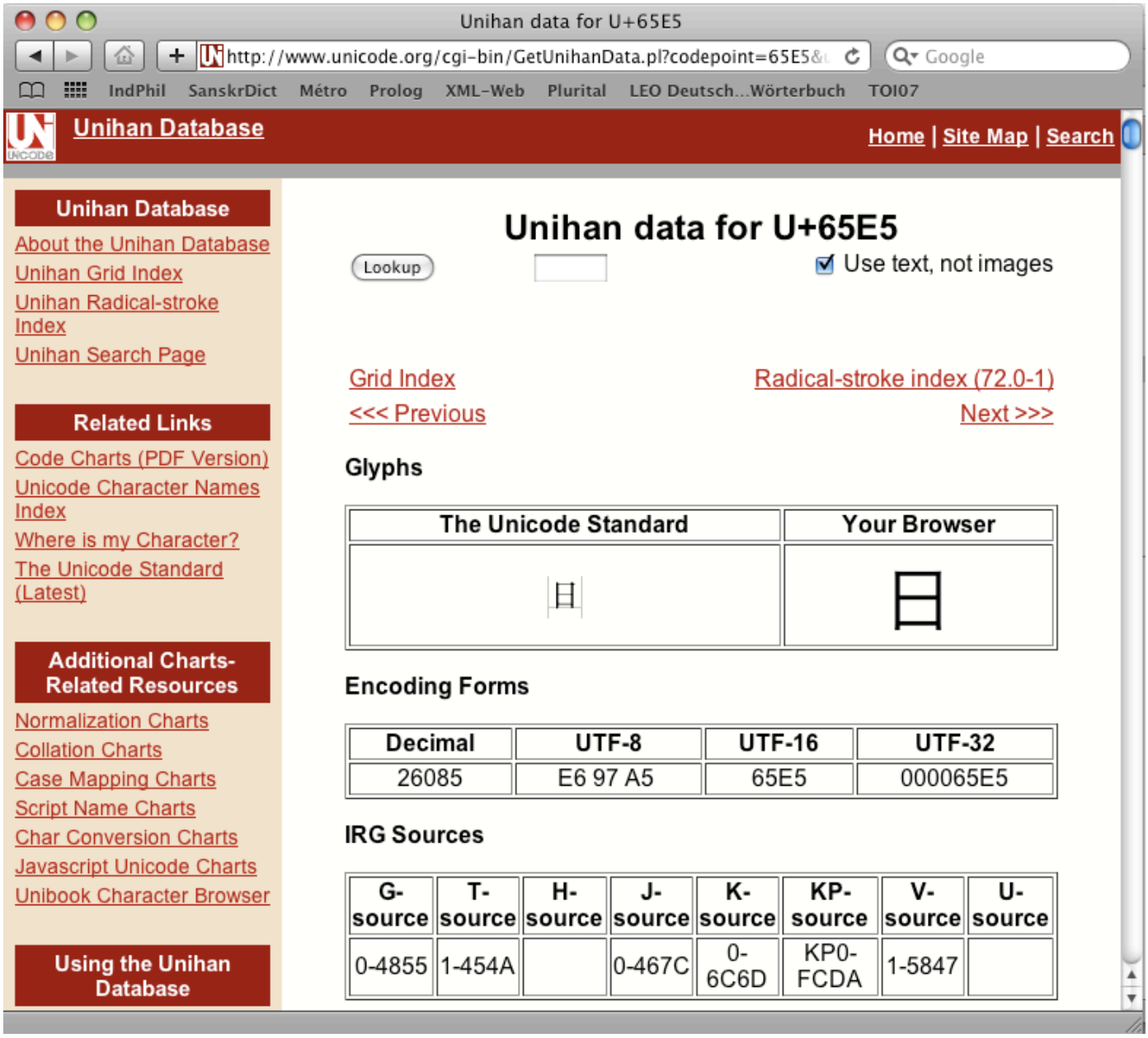
4. **Illustrations**

On pourrait les multiplier !
Je me bornerai à un exemple de logiciel exploitant le catalogue Unicode vu comme une base de donnée et à un requête sur la sous-base Unihan qui concerne les caractères chinois.

Une image du logiciel *Unicode Checker*, sur MacOS-X.



Une requête à Unihan.
La forme générale de la requête est
`http://www.unicode.org/cgi-bin/GetUnihanData.pl?codepoint=`*`lenumérohex`*`[&useutf8=true]`
Voici le début de la réponse pour le caractère *soleil* : la position de l'ascenseur à droite de l'image donne un idée de sa longueur totale !



Exemples de conséquences pratiques

1. **L'adoption d'UTF-8 comme codage par défaut pour XML**

   Pendant longtemps, le seul système de codage reconnu internationalement et resté ASCII.
   Pour écrire des pages Web en français, par exemple, il fallait donc coder les lettres accentuées ('é' = "`&eacute;`").
   Ensuite, cette position a été prise par Latin-1, ce qui oblige à coder tous les autres jeux de caractères.
   Enfin, le consortium W3G a promulgué UTF-8 comme codage par défaut pour les fichiers XML, c'est-à-dire qu'UTF-8 est automatiquement adopté en l'absence de mention contraire.

   Comme les pages Web modernes sont écrites dans le "dialecte" XHTML de la "langue" XML, il faut désormais refuser explicitement d'être international...

   Les pages de Wikipédia sont ainsi internationales "par construction" ! Exemple :

Pour d'autres sites, UTF-8 est un des choix de codage possibles :
ainsi, par exemple *Titus* où on trouve une collection très complète de textes anciens (`http://titus.uni-frankfurt.de/`)

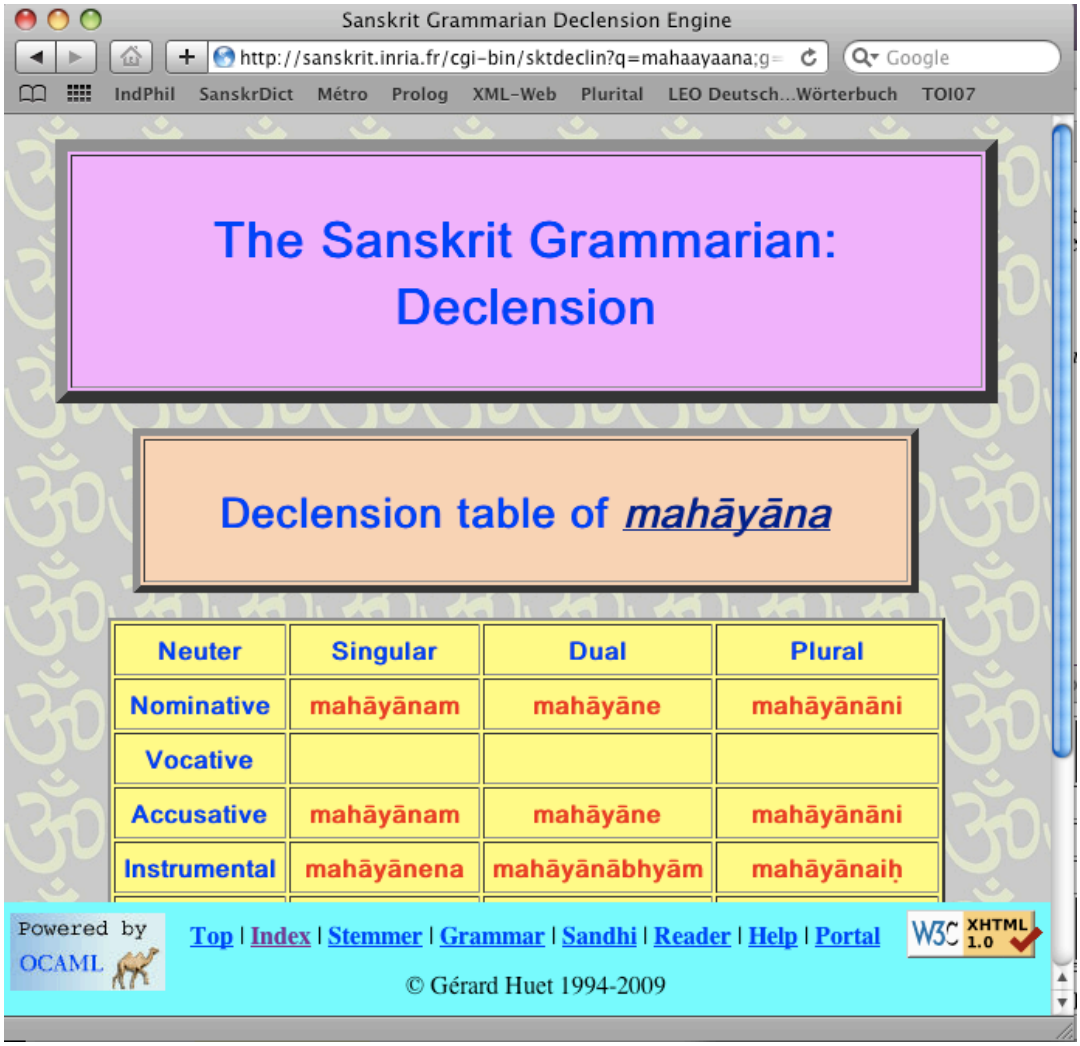
2. **L'internationalisation des URI**

   Les URI (*Uniform Resource Identifier*) sont les noms employés dans le communications sur Internet pour désigner le objets du discours. Ces noms, par un privilège exorbitant, sont censés avoir une interprétation unique à l'échelle planétaire. Jusqu'ici, on ne pouvait les écrire qu'en ASCII.
   Le W3C a récemment décidé que tous les caractères Unicode seraient acceptables.

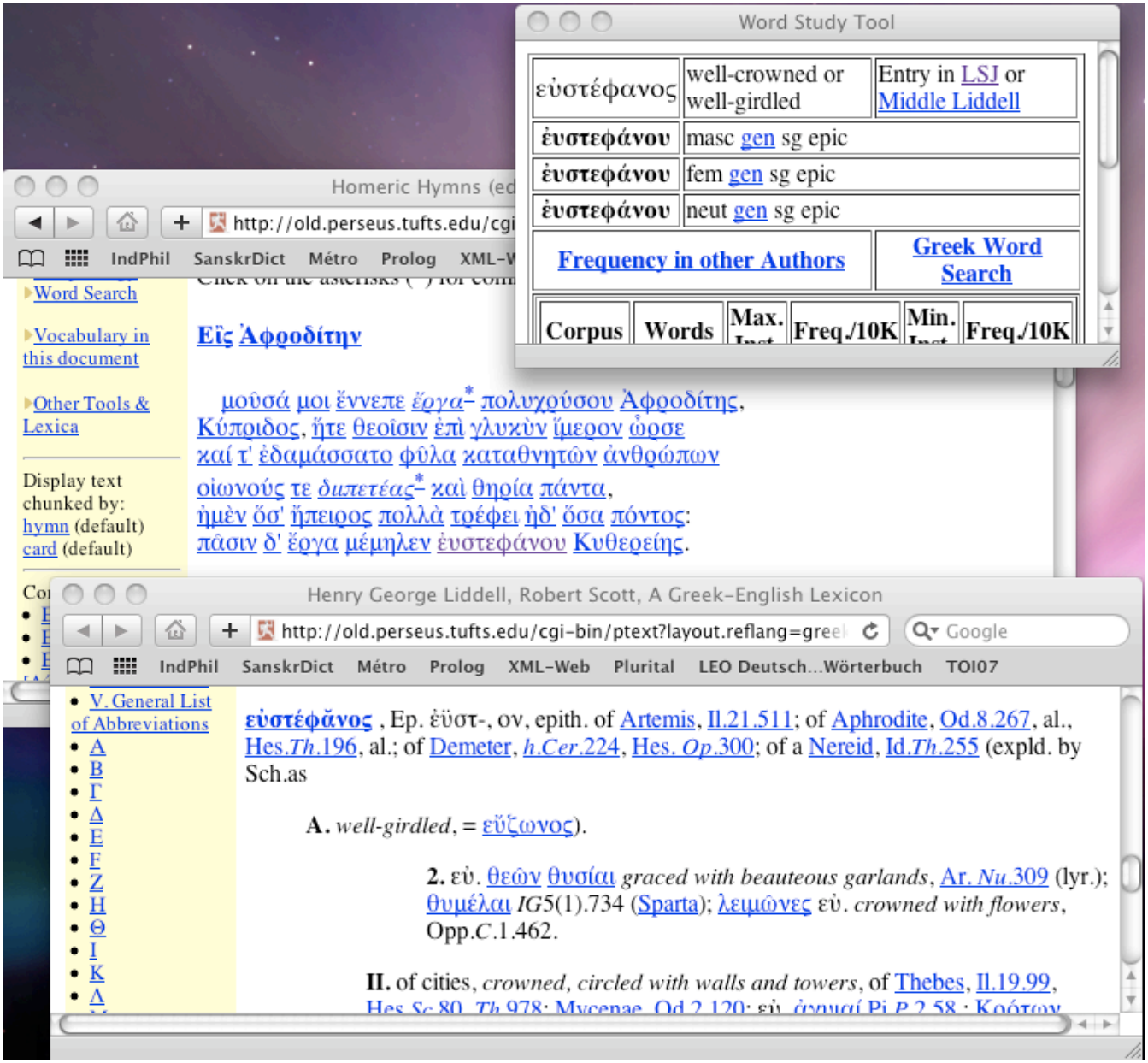
   Par exemple, on peut valablement désigner la page de Wikisource consacrée à la *Śvetāśvatara Upaniṣad* par
   `http://wikisource.org/wiki/श्वेताश्वतर_उपनिषद्`

3. **Deux exemples d'aide à la lecture des textes anciens**

   - Le site de Gérard Huet à l'INRIA pour le sanskrit (`http://sanskrit.inria.fr/`) propose une aide grammaticale en ligne :



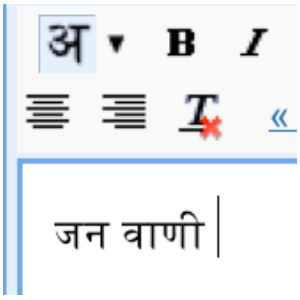
- Le site Perseus (`http://www.perseus.tufts.edu/hopper/`) pour le grec offre un analyseur morphologique assorti de l'accès direct au dictionnaire de Liddell & Scott.



4. **Intégration d'un utilitaire de translittération dans *GMail***

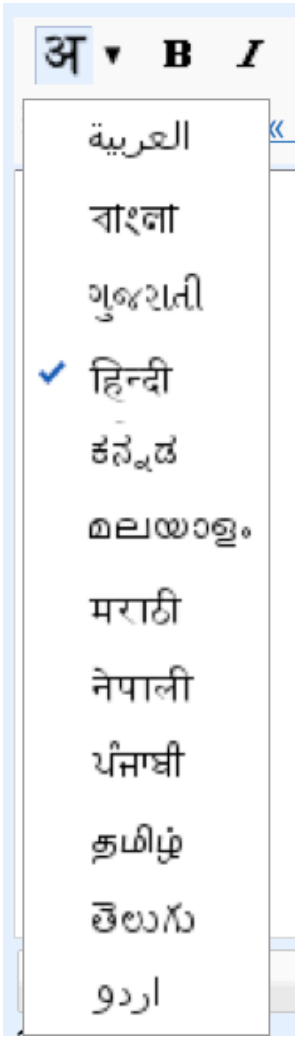
Puisque Google a été invoqué dans le résumé de cet exposé, illustrons le rôle d'Unicode dans le service de courrier *GMail*.
L'utilisateur peut écrire dans une langue indienne, avec l'alphabet *ad hoc*, en tapant des caractères latins mais sans pour autant connaître la translittération officielle : il suffit d'une approximation, la machine reconnaît le mot visé.
Cette opération qui met évidemment en jeu une certaine connaissance de la langue, se fait sur le serveur, mais le texte produit se trouve côté client - en UTF-8 !

Exemple en hindi



L'utilisateur a tapé "`jan vani `". Chaque espace a provoqué la translittération en devanâgari.
Noter que la forme officielle est "`jan vāṇī`", avec `ā` long, `ī` long et `ṇ` rétroflexe.
La réalisation normale de "`jan vani`" serait "जन वनि".
Mais la machine (en l'occurrence le serveur de Google, pas la machine de l'utilisateur) a compris qu'il s'agissait de "*la voix du peuple*" et elle rétabli l'orthographe correcte !

Le choix de la langue dans laquelle on veut écrire (à condition d'avoir une police convenable sur sa machine) est assez étendu à nos yeux, mais il ne couvre pas encore les 22 langues officielles de l'Union indienne !



arabe

bengali (bangla)

gujarâti

hindi

kannada

malayâlam

marâthi

népâli

panjâbi (en écriture gurmukhi)

tamoul

télougou

ourdou