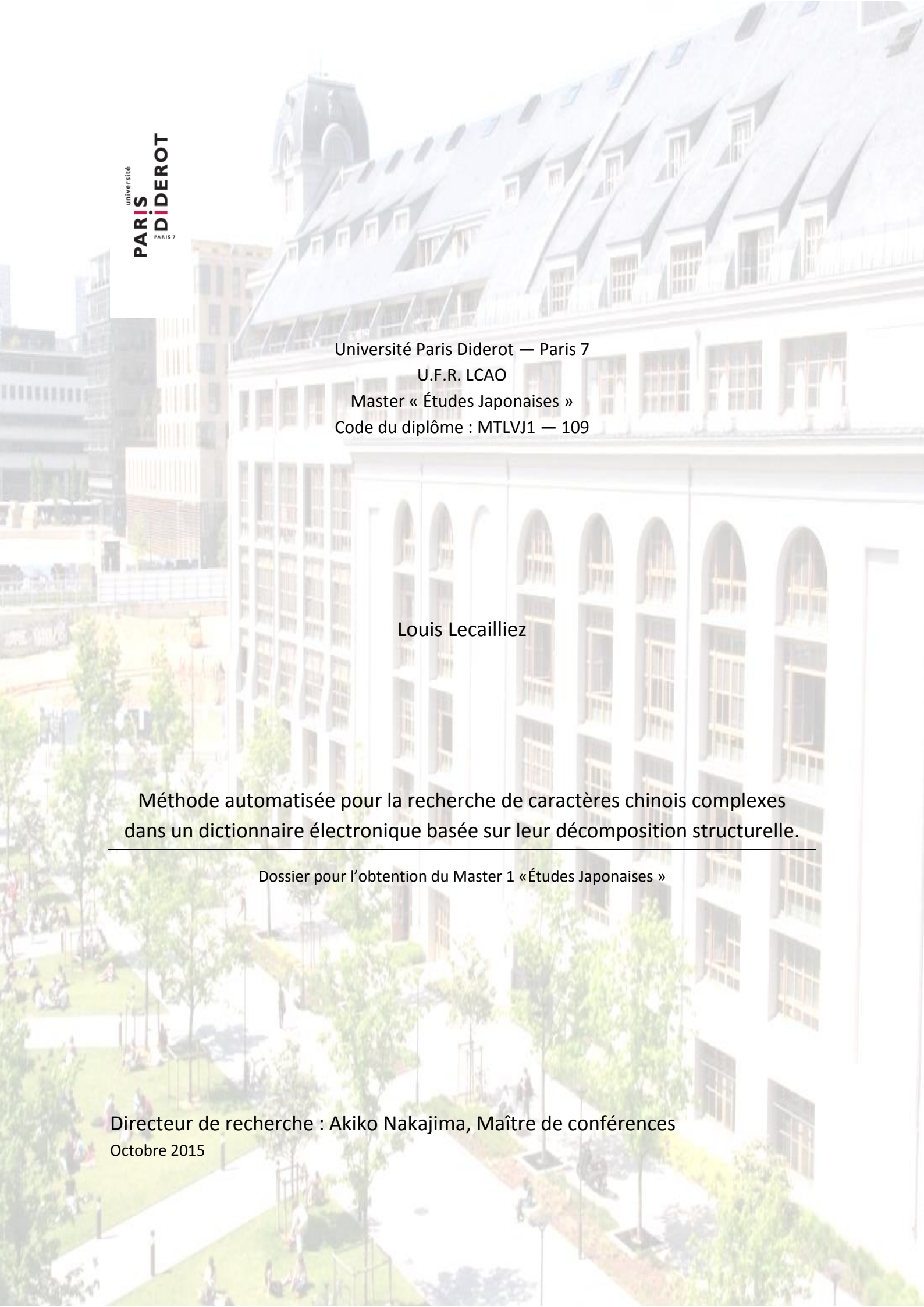


Université Paris Diderot — Paris 7
U.F.R. LCAO
Master « Études Japonaises »
Code du diplôme : MTLVJ1 — 109

Louis Lecailliez

Méthode automatisée pour la recherche de caractères chinois complexes dans un dictionnaire électronique basée sur leur décomposition structurelle.

Dossier pour l'obtention du Master 1 «Études Japonaises »

Directeur de recherche : Akiko Nakajima, Maître de conférences

Octobre 2015

1 Table des matières

2 Conventions utilisées dans ce mémoire 5
2.1 Forme des sinogrammes 5
2.1.1 Polices de caractères utilisées 5
2.2 Code de langue 5
2.3 Transcriptions 6
3 Introduction 7
3.1.1 Plan du travail 7
4 Recherche dans un dictionnaire électronique 8
4.1 Définitions 8
4.1.1 Avant-propos 8
4.1.2 Mot cible et zone d'entrée 8
4.1.3 Recherche directe, indirecte et résultats 8
4.1.4 Méthode d'entrée 10
4.1.5 Ordinateurs sans méthode d'entrée tactile 10
4.1.6 Autre cas d'usage 11
4.2 Description orale d'un mot 11
4.2.1 Mot écrit en kanji 11
4.2.2 Description orale d'un caractère chinois 11
4.2.3 Application informatique 12
4.3 Pertinence de l'intégration à l'existant 13
4.3.1 Dictionnaires électroniques utilisées 13
4.3.2 Tagaini Jisho 13
4.4 Taigaini Jisho 14
4.4.1 Présentation de l'interface 14
5 Structure composée des sinogrammes 17
5.1 Définitions 17
5.1.1 Logogramme 17
5.1.2 Écriture logographique 18
5.1.3 Clef et Composante dans l'écriture chinois 18
5.2 Homonymes dans les écritures logographiques 18
5.2.1 Comparaisons entre écritures logographiques 18
5.2.2 Cas de l'écriture sumérienne 18
5.2.3 La solution chinoise à l'homonymie 19

5.2.4 Autre originilité de l'écriture chinoise 20
5.2.5 Utilisation pour la langue japonaise 21
5.3 Décompositions informatisées 21
5.3.1 Données de décompositions 21
5.4 Unicode et IDS 21
5.4.1 IDC 22
5.4.2 Arité des opérateurs et forme préfixe 22
5.4.3 IDS 23
5.5 Kanji Database Project 24
5.6 Entités CDP 25
5.7 Problèmes des décompositions IDS 25
5.7.1 Ambiguïté des décompositions 25
5.7.2 Points de code multiples 26
5.8 Licences 26
5.8.1 Gnu General Public Licence (GPL) 26
5.8.2 Creative Commons 26
6 Algorithme de recherche 28
6.1 Fonctions nécessaires 28
6.1.1 Recherche de sinogrammes comprenant une composante c 28
6.1.2 Décomposition d'un sinogramme k en ses composantes 28
6.1.3 Existence d'une composante c dans un sinogramme K 28
6.2 Algorithme formel 28
6.3 Seconde formulation de l'algorithme 29
6.4 Description de l'implémentation 29
6.4.1 Structure du fichier de données IDS 29
6.5 Utilisation du fichier IDS.TXT 30
6.5.1 Description non terminale 30
6.6 Représentation sous forme de graph 30
6.6.1 Arbre de décomposition 30
6.6.2 Arbre de composition 31
6.6.3 Graph de données 31
6.7 Implémentation des fonctions r_c et d à partir de graphs 32
6.8 Problèmes rencontrés dans l'implémentation 33
6.8.1 Aperçu du problème 33

6.8.2 Décompositions problématique 33
6.9 Code source 34
6.10 Application pratique 35
6.10.1 Workflow de recherche d'un kanji complexe sur une montre connectée 35
6.10.2 Écran A : tracé 36
6.10.3 Écran B : sélection d'un caractère de recherche 36
6.10.4 Écran C : sélection du résultat 37
7 Conclusion 38
8 Références v
8.1 Articles v
8.2 Livres v
8.3 Thèses vi
8.4 Pages web vi
8.4.1 Base de données de kanji vi
8.4.2 Dictionnaires vi
8.4.3 Organisations vi
8.4.4 Applications web ou téléchargement vii
8.5 Logiciels vii
8.6 Brevets vii

2 Conventions utilisées dans ce mémoire

2.1 Forme des sinogrammes

Ce travail porte sur les caractères chinois qui sont les unités de base de l'écriture chinois. Cette écriture a été utilisée dans une large aire géographique et durant une période s'étendant de plusieurs millénaires à la fois pour écrire « le chinois » mais aussi des langues vernaculaires de peuples différents.

Les peuples de langues non siniques ont, lors de l'adaptation de ce système d'écriture à leur langue, créées de nouveaux caractères qui ne se retrouvent pas dans les autres langues de la sinosphère. Dans ce domaine, c'est le vietnamien qui a été le plus prolifique avec la création de milliers de Chữ Nôm (𡨸喃). Au Japon, les caractères créés localement sont nommés Kukoji (国字).

Le présent travail s'attache à proposer une méthode de recherche reposant sur la structure des sinogrammes. Comme les *chu nom* et les *kukoji* respectent cette structure dans leurs inventions, il est tout à fait applicable à ceux-ci. Puisque nous sommes dans le cadre d'un master d'études japonaises, les exemples donnés proviendront principalement du japonais.

Toutefois, comme les caractères traditionnels sont communs aux quatre langues est-asiatiques faisant ou ayant fait usage de sinogrammes (chinois, coréen, japonais, vietnamien), et que ceux-là sont plus intéressants que les formes simplifiés (新字体 *shinjitai* japonais, 簡體字 *jiantizi* chinois) au niveau de leur composition, ils seront préférés dans les exemples et dans la réalisation informatique afin d'être plus accessible aux spécialistes des langues autres que le japonais.

2.1.1 Polices de caractères utilisées

Par défaut, c'est la police *PMingLiU* qui est utilisée pour les caractères chinois entrée dans le texte. Comparée à une police créée pour le japonais telle que *MS Mincho*, cette police prévue pour l'affichage du chinois traditionnelle supporte un plus grand nombre de caractères.

La police *PMingLiU-ExtB* est utilisée pour l'affichage d'un caractère appartenant au bloc Unicode *CJK Unified Ideographs Extension B*.

Dans les cas où une meilleure lisibilité des caractères est requise (c'est le cas dans la partie qui décrit l'algorithme), c'est la police de Microsoft *Meiryo UI* qui est utilisée.

2.2 Code de langue

Si le mot existe en japonais et en chinois mandarin, il sera présent sans aucune glose dans le texte. N'ayant pas de connaissance du coréen et étant donné la plus faible utilisation des caractères chinois dans ce contexte, l'existence de ces termes dans cette langue ne sera pas vérifiée. De même pour le vietnamien.

Dans le cas où un terme n'existerait qu'en japonais ou qu'en mandarin et que le contexte ne permet pas de déduire la langue du mot présenté, ou qu'il faille préciser pour quelque raison que ce soit la langue d'un mot, la langue de celui-ci sera précisée par un code de langue[1] souscrit suffixé.

Les codes de langues utilisés sont :

Code	Langue
fr	français
en	anglais
jp	japonais
zh	chinois mandarin (contemporain)

Ainsi le mot $chat_{fr}$ est à comprendre comme un félin tandis que $chat_{en}$ désigne un dispositif de communication textuelle électronique.

2.3 Transcriptions

N'ayant pas l'utilité d'un système qui transcrive précisément la morphologie de la langue japonaise dans le présent travail, le système de transcription du japonais utilisé sera le système Hepburn.

Pour les transcriptions transcription du chinois et du chinois classique c'est le système pinyin qui sera utilisé, et ceci dans un souci d'accessibilité. La première fois qu'un terme chinois cité, sa transcription est indiquée en pinyin avec tous ses diacritiques accentuelles. Si cette transcription est ensuite réutilisée, elle peut être indiquée sans accent. Si la transcription est donnée en chinois et en japonais, la transcription japonaise est mise en premier.

[1] Codes extrait de la norme ISO 639-1.

3 Introduction

Par rapport à la recherche d'un mot écrit alphabétiquement, la recherche d'un caractère chinois dans un dictionnaire électronique pose problème. La recherche de caractère complexe est généralement implémentée de manière moyennement satisfaisante pour l'utilisateur final : il doit avoir mémorisé le numéro des radicaux Kangxi[2] ou procéder à de multiples étapes manuelles afin de trouver ce qu'il cherche.

Je pense qu'il est possible d'implémenter une méthode de recherche de sinogrammes complexes qui se base exclusivement sur le clavier ; et qui a donc un avantage de vitesse sur les interfaces utilisant la souris. En outre une telle méthode pourrait permettre de chercher facilement des caractères non supportés dans les méthodes d'entrées japonaises et chinoises existantes.

3.1.1 Plan du travail

Dans la première partie (0

[2] Association Ricci (2010).

Recherche dans un dictionnaire électronique) nous nous attacherons à définir quelques éléments liés à la recherche dans un dictionnaire électronique, qui diffère sensiblement d'une recherche dans un ouvrage papier. Nous mettrons en lumière certains points faibles des interfaces utilisateurs (UI) existantes avec pour exemple un dictionnaire de japonais et de kanji open-source et gratuit. Puis l'idée sous-jacente à la méthode proposée sera expliquée en langue naturelle (français) et illustrée à l'aide d'exemples concrets liés à la vie de tous les jours.

Dans la seconde partie (0

Structure composée des sinogrammes), nous aborderons la problématique de la structure des caractères chinois, qui est un point théorique important pour la réalisation technique. Je proposerai également une hypothèse sur l'origine de cette structure, en faisant un parallèle avec d'autres écritures logographiques.

C'est également dans cette partie que serons listées et décrites les ressources techniques (bases de données de caractères et leurs décompositions) qui pourront être réutilisées pour une mise en œuvre informatique de la méthode proposée dans ce travail.

Enfin l'algorithme lui-même sera décrit de manière formelle, avec quelques réflexions sur la manière dont il pourrait effectivement être implémenté, avec en tête l'idée des performances suffisamment bonnes pour une utilisation sur mobile.

4 Recherche dans un dictionnaire électronique

4.1 Définitions

4.1.1 Avant-propos

Afin que le propos soit compréhensible, il convient de définir les termes qui seront utilisés dans ce travail. Les définitions données ici sont *ad hoc* : elles ne reflètent pas l'utilisation de termes spécifiques dans le champ de la lexicologie.

En outre, elles sont données sous un angle abstrait et fonctionnel et ne s'occupent en aucune sorte des détails techniques d'implémentation.

4.1.2 Mot cible et zone d'entrée

Nous appellerons dans la suite de cette étude « mot cible » ou « sinogramme cible » le sinogramme ou le mot dont on veut rechercher une entrée possiblement existante dans un dictionnaire informatisé.

On prend pour hypothèse que ce dictionnaire est munie d'une zone d'entrée (ou champ de recherche) qui permet de saisir du texte.

4.1.3 Recherche directe, indirecte et résultats

On appelle résultats la liste des entrées du dictionnaire obtenue après une recherche. On appelle recherche directe une recherche pour laquelle on écrit le mot cible dans la zone d'entrée et où on attend une entrée correspondante dans la liste des résultats.

Figure1 : recherche directe.

Une recherche indirecte est une recherche qui n'est pas directe. Par exemple dans le cas d'un dictionnaire de caractères chinois, une recherche par romanisation (peu importe la méthode de romanisation proprement dite) est une recherche indirecte.

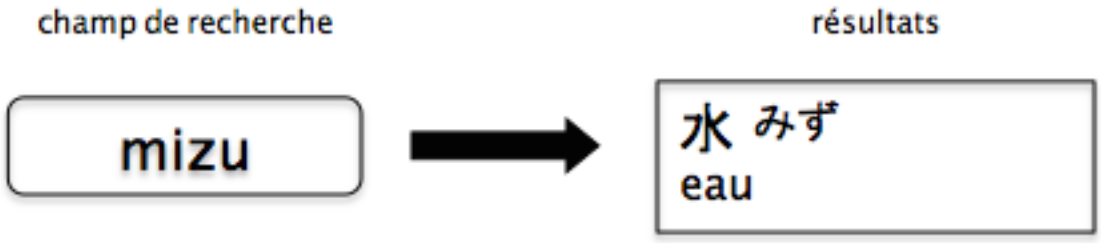


Figure 2 : recherche indirecte.

Dans l'exemple de recherche indirecte, on voit que la différence tient au fait que la recherche a été effectuée sur un autre champ de données que le champ principal (水). En outre, une transformation a été appliquée avant la recherche : la romanisation a été transformée en interne en *kana* et c'est le résultat de cette recherche qui a été retournée.

Dans le cas d'un dictionnaire de français, une recherche directe se fait en écrivant dans le champ de recherche les lettres qui composent le mot leur ordre d'apparition. Des traitements supplémentaires peuvent devoir être fait par l'utilisateur humain avant d'obtenir des résultats : par exemple mise au singulier du terme recherché ou mise à la forme infinitive.

Un dictionnaire peut effectuer toutes ou parties de ces opérations et retourner des résultats ne correspond pas exactement au contenu de la zone d'entrée. Des techniques dites de *fuzzy match* sont alors utilisées.

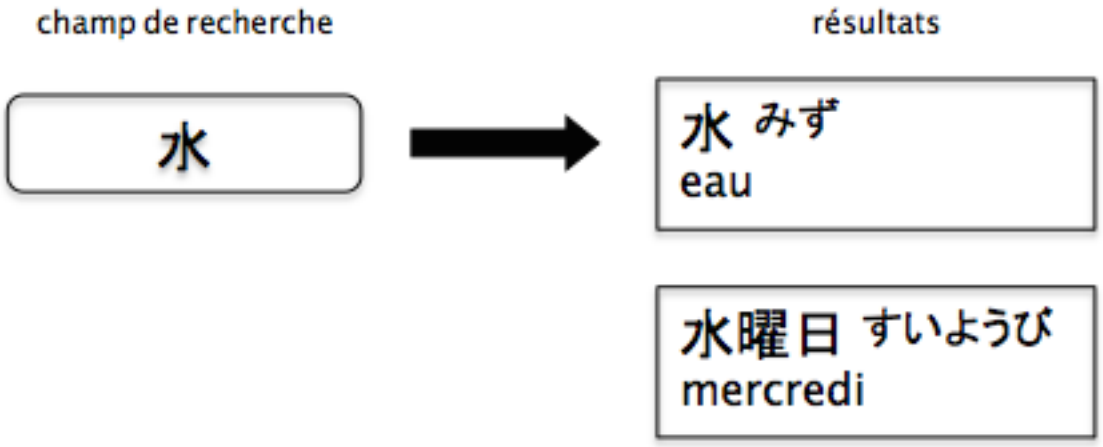


Figure 3 : recherche avec *fuzzy match*.

Dans l'exemple de la figure 3 la recherche renvoie également des entrées qui commencent par le caractère de l'eau (水) en plus des entrées qui correspondent exactement à ce sinogramme.

4.1.4 Méthode d'entrée

Dans le cas d'un caractère chinois, écrire le caractère dans la zone d'entrée ne tient pas de soi car aucun clavier ne comporte directement de touches correspond aux caractères chinois. Il faut passer par une méthode d'entrée (IME).

Une IME permet de convertir une séquence d'appui de touches (touches de lettres, *dead keys*, touches spéciales, etc.) en une séquence de texte dans une écriture donnée.

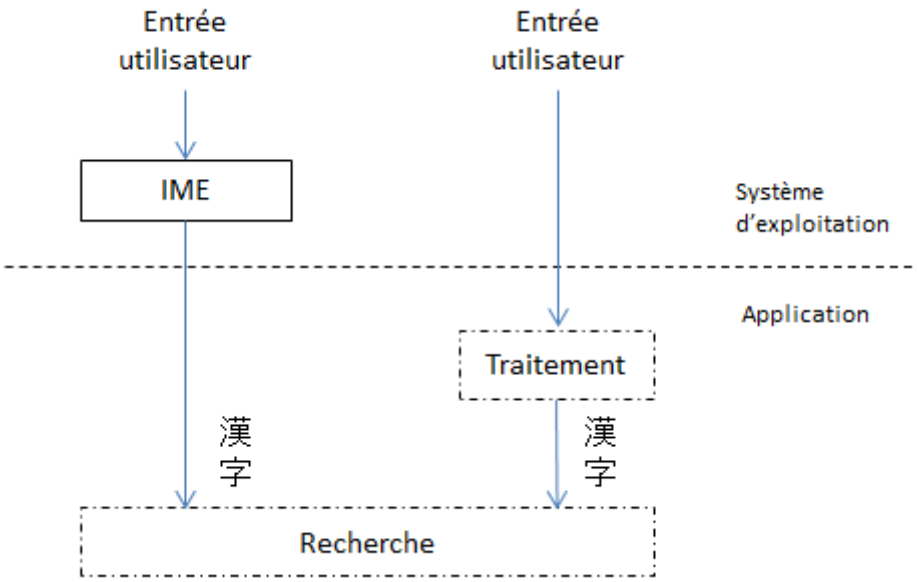


Figure 4 : comparatif entre l'utilisation ou non d'une IME avec un dictionnaire.

Dans le cas général, une méthode d'entrée sert à écrire du texte non alphabétique. Dans le cas de la recherche dans un dictionnaire, certaines des responsabilités fonctionnelles sont partagées entre une IME et le dictionnaire. En effet, si l'IME permet d'entrer un caractère chinois, qui est ensuite utilisé pour une recherche directe, le dictionnaire n'a pas besoin de s'en occuper.

Dans le cas contraire, le code applicatif du dictionnaire doit pouvoir convertir une entrée utilisateur (suite de lettres, tracé, etc.) en caractères chinois, si cela est nécessaire, afin de pouvoir effecteur l'opération de recherche demandée.

La programmation d'une IME par rapport à l'implémentation d'une méthode spécifique à chaque application à de nombreux avantages. Citons notamment une expérience utilisateur plus cohérente et une grande économie de moyens.

Mais si cela n'est pas possible, par exemple parce que le système sous-jacent n'en est pas équipé ou que l'utilisateur ne sait ou ne peux pas s'en servir, c'est au dictionnaire que reviendra la responsabilité de fournir une méthode permettant de chercher le mot cible parmi ses entrées.

4.1.5 Ordinateurs sans méthode d'entrée tactile

En particulier, si les méthodes d'entrées tactiles couplées à un logiciel de recherche manuscrite sont particulièrement efficaces pour entrer des caractères inconnus de l'utilisateur, elles ne sont pas

disponibles partout : ordinateurs de bureau et pc portables à *trackpad* de petite surface. Ces machines disposent toutefois d'IME qui tirent profit du clavier.

C'est à ce cas là qu'est consacré ce dossier : il s'agit de fournir une méthode de recherche indirecte pour des caractères chinois complexes sur des ordinateurs pourvus d'IME textuels et dépourvus d'IME tactiles.

La méthode en question sera donc décrite d'un point de vue fonctionnel, puis les conditions techniques nécessaires à sa réalisation seront analysées.

4.1.6 Autre cas d'usage

Au moins un autre cas où l'application de la méthode peut être intéressante existe: celui d'un appareil muni d'une IME (reconnaissance manuscrite ou non) pour le chinois et la recherche de Chữ Nôm[3]. Cette dernière écriture ayant été abandonnée, elle n'est plus lisible que par une centaine de spécialistes d'après la Fondation Vietnamienne pour la préservation des Nôm (VNPF 2015). Il n'y a donc généralement pas d'IME fournie par les fournisseurs de système d'exploitation, ce qui obligerait le créateur d'un dictionnaire électronique de Nôm à implémenter lui-même une méthode de recherche dans celui-ci, avec toutes les difficultés que ça comporte.

Le méthode proposée ici est cependant applicable et permet donc d'implémenter facilement la recherche de caractères non compris dans les IME standards à l'aide d'une telle IME. Bien sûr, ce n'est pas seulement la réutilisation du principe qui est applicable, mais le partage même de code informatique qui pourrait l'être sans avoir à tout reprogrammer de zéro.

4.2 Description orale d'un mot

4.2.1 Mot écrit en kanji

La méthode proposée s'inspire de la méthode orale pour décrire un mot écrit en kanji ou un caractère. On indique une suite de mot ou de noms de préférence connus ou courants qui les contiennent afin d'indiquer sans ambiguïté les kanji utilisés dans ce nom décrit.

Un exemple en japonais, généré automatiquement par un site internet (MasanoriSoft, 2008) pour le nom 貴之 (Takayuki) : « 貴重の 貴 に、芥川龍之介(あくたがわりゅうのすけ)の 之です。 » c'est-à-dire en français « C'est le *ki* de *kichō* (précieux) et le *no* d'Akutagawa Ryūnosuke. »

La méthode fonctionne également en chinois de la même manière : pour épeler la transcription non standard 陸逸 (lùyì) du prénom Louis, on peut utiliser la proposition suivante : « 大陸的陸，逸樂的逸 » qui équivaut à « le [caractère] lù de continent et le yì de plaisirs ».

4.2.2 Description orale d'un caractère chinois

Des sinogrammes individuels peuvent également être explicités en énonçant un mot connu qui le contient. Cependant, dans le cas d'un caractère rare il se peut qu'aucun mot ne soit suffisamment connu (fréquent) pour que cela fonctionne sans accros. Dans ce cas, il ce sont les composantes (部首, bushu, bùshǒu) du caractère qui sont énoncés.

---

[3] Cela n'est pas rhétorique : je réfléchis à la numérisation d'un certain dictionnaire annamite-français et à la création d'une application mobile pour les données de ce dictionnaire.

Ces composantes possèdent un nom particulier en fonction de leur tracé et de leur position dans l'ensemble du caractère.

Les locuteurs non natifs tels que les japonisants n'ont, par constat personnel, pas connaissance de ces noms (ou alors trop limités pour un usage efficient). Ils se servent donc de formule un peu plus longue, dans leur langue d'origine.

Le caractère 弑 (shi, shì : régicide, patricide) peut ainsi se décrire à un autre francophone japonisant par la phrase suivante : « Ce sinogramme a pour partie gauche la partie gauche du kanji *satsu* de *ansatsu*, l'assassinat et pour partie droite le kanji de *shiki* la cérémonie ».

4.2.3 Application informatique

Cependant, ce type description qu'il est possible d'utiliser à l'oral n'est pas du tout envisageable telle quelle pour une recherche dans un dictionnaire : le contenu textuel d'entré est beaucoup trop long et nécessite un traitement de la langue naturel qu'il n'est pas facile de mettre en place.

Notons que c'est techniquement possible. L'état de l'art du traitement automatique des langues gère très bien ce type de problème de texte lié à un domaine spécifique. Mais c'est tout simplement trop long et coûteux à mettre en place dans un dictionnaire, pour au final un usage marginal et une expérience utilisateur rebutante.

Il est toutefois possible de s'inspirer de cette méthode orale, mais plutôt que faire préciser par l'utilisateur quelle partie de quel caractère est utilisée dans le sinogramme cible, on peut tenter de déléguer ce travail à la machine.

De cette façon, la séquence textuelle à écrire dans le champ d'entrée d'un dictionnaire pour recherche le caractère 弑 se simplifie en « 殺式 ». Pour ne pas contraindre un ordre spécifique qui nécessiterait d'être appris par l'utilisateur et à appliquer à chaque utilisation de la méthode, on considère les permutations des caractères utilisés en entrée comme équivalente. Spécifier cela permet également à l'algorithme de réorganiser l'ordre d'entrée des sinogrammes si besoin est.

Ainsi « 殺式 » et équivalent à «式殺 » et devrait idéalement produire la même liste de résultats.

La figure ci-dessous illustre l'idée détaillée plus haut.

Figure 5 : recherche par la méthode exposée dans ce dossier

4.3 Pertinence de l'intégration à l'existant

Maintenant que le sujet a été explicité et expliqué dans ses grandes lignes penchons-nous sur l'utilité concrète de sa mise en œuvre dans un dictionnaire informatisé. Pour cela nous étudierons le *workflow* de la recherche dans deux dictionnaires électroniques pour pc : le dictionnaire japonais-anglais ou japonais-français Tagaini Jisho et le dictionnaire de chinois-français Le Grand Ricci Numérique.

4.3.1 Dictionnaires électroniques utilisées

4.3.2 Tagaini Jisho

Le premier est un dictionnaire de langue japonaise et de kanji, libre[4] et gratuit et disponible pour plusieurs systèmes d'exploitation. Le mot « libre », dans un contexte logiciel, signifie que son code source (les fichiers qui permettent de créer le programme lui-même) sont disponibles pour tous les utilisateurs qui en font la demande et qu'ils peuvent s'en servir pour créer des logiciels dérivés, à condition de respecter certains engagements tel que le fait ce second logiciel soit lui-même libre.

Les données lexicographiques proviennent de plusieurs sources dites ouvertes qui permettent une réutilisation, notamment commerciale, à condition d'en spécifier l'origine dans le logiciel. Il utilise entre autres les fichiers bien connus JMDict et Kanjidic développés à l'origine par Jim Breen de l'université de Monash. Un troisième fichier, KanjiVG, sera décrit plus en détails dans la partie 3 de ce mémoire.

Techniquement, il est programmé en C++ en se basant sur le *framework* Qt, et utilise une base de données SQLite pour stocker et chercher parmi ses données.

[4] Placé sous licence GNU General Public Licence version 3.0.

4.4 Taigaini Jisho

4.4.1 Présentation de l'interface

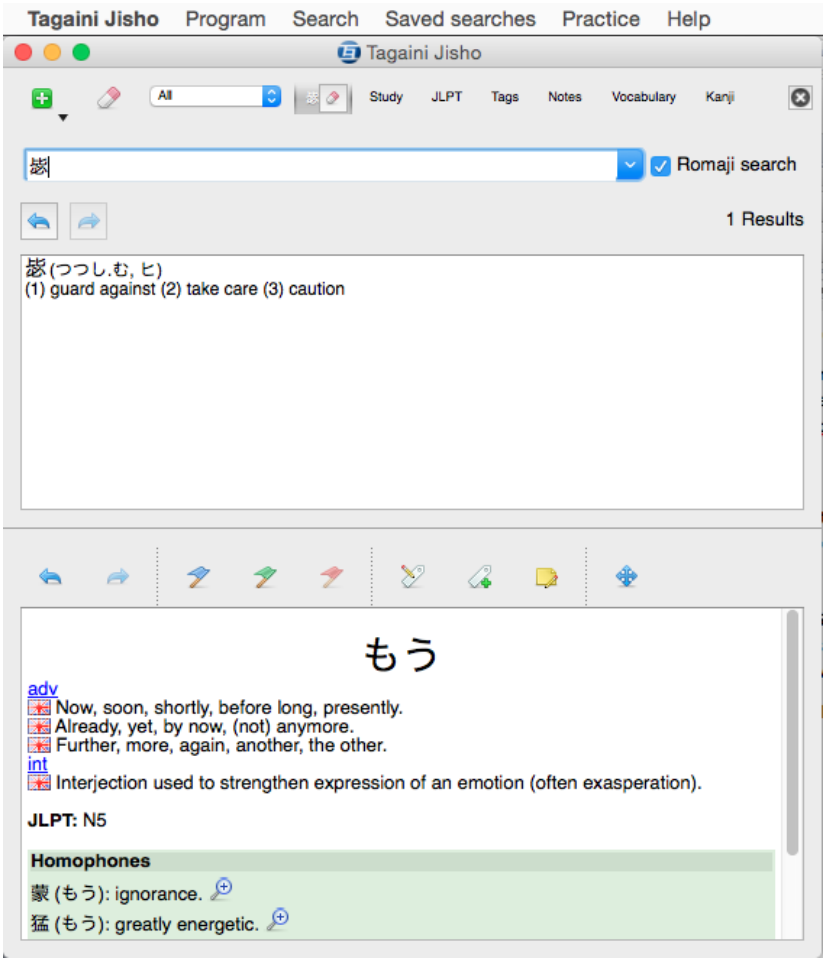


Figure 6 : interface principale de Tagaini Jisho

Voici une capture d'écran de Tagaini Jisho sur Mac OS. Comme ce système d'exploitation sépare la barre de menus des fenêtres d'un programme[5] mais que celle-ci est indispensable à la compréhension des étapes permettant de chercher un caractère, j'ai regroupé la fenêtre du programme et la barre de menu de façon à ce que les deux apparaissent sur la capture d'écran.

De haut en bas, on trouve la barre du menu, qui appartient à la fenêtre sur Windows. Puis la barre de titre, qui ne contient que le titre, le logo et des boutons gérés fournis par le système d'exploitation pour la gestion de la fenêtre.

Immédiatement en dessous ce situe la zone dédiée à la recherche. Le champ d'entrée textuel où se trouve le caractère 慥 (tsutsushi.mu, bì : se prémunir) correspond fonctionnellement à la zone d'entrée que nous avons définie dans la première partie. Du texte sous de multiples formes peut y être insérer afin d'effectuer des opérations de recherche : mot japonais (sous forme de kanji, de kana, d'un mélange des deux ou romanisé), sinogramme, mots en anglais.

[5] Ce que j'ai personnellement toujours trouvé anti-ergonomique au possible.

Afin de faire la distinction en recherche d'un mot anglais et recherche d'un mot japonais romanisé, il existe à droite de ce champ de recherche une case (*checkbox*) permettent d'indiquer au programme de quel type de recherche il s'agit. Si elle est cochée, les entrées dont une prononciation correspond à la romanisation seront ajoutées aux résultats de recherche.

Plus haut se trouve différentes options qui permettent d'effectuer des recherche plus précises (par exemple en filtrant les résultats par niveau de JLPT ou en sélectionnant des résultats qui correspondent à des mots / sinogrammes) ou de profiter de certaines fonctionnalités supplémentaires du programme notamment liés à la constitution de liste de vocabulaire et à leur apprentissage.

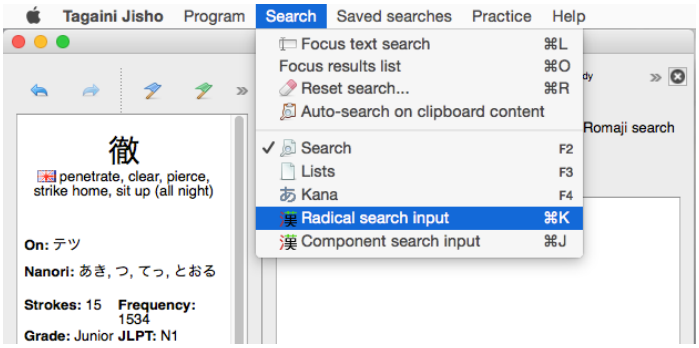


Figure 7 : recherche par radical et par composantes

En plus des options de recherche de sinogrammes présentées dans la capture ci-dessous, il existe deux boites de dialogues nommés « Radical Search Input » et « Component Search Input » qui permettent des rechercher des caractères complexes à partir de leur clef et ou de composant.

Figure 8 : boite de recherche par radical après sélection d'une clef

Le fonctionnement de ces deux boites de dialogues est similaire : une liste en deux dimensions de composantes graphiques est présentés et un clic sur l'un d'elle entraîne (1) la réduction du nombre de composantes dans la boite de dialogue et (2) l'apparition ou la réduction d'une liste de caractères possibles comprenant ces composants.

Ces boites sont relativement bien faites, mais pour les avoir utilisés plusieurs années, je leur reproche certains choses :
(a) la séparation en deux boites différentes pour les composantes non clefs et les clefs
(b) le changement des composantes proposées ne se fait qu'en réduction.

Le point (a) fait qu'il est parfois nécessaire, lorsqu'on en sait pas si une composante est une clef ou non, ou si l'on se trompe, de parcourir ces deux boites de dialogue, ce qui est une perte de temps.

En outre si la séparation des deux est pertinente d'un point de savant (c'est à dire de l'organisation dans les dictionnaires papier de référence) elle ne l'est plus dans le cadre d'un dictionnaire électronique où le manque de place qui impose une limite aux nombres de clefs n'est plus pertinente : le fait qu'une composante soit une clef ou non devrait être transparent à l'utilisateur qui ne connaît pas cette information puisqu'il est possible techniquement possible d'indexer un caractère par toutes les composantes qui le constituent. D'ailleurs certains dictionnaires papier indexent certains caractères dont la clef peut porter à confusion sous plusieurs clefs ; il ne s'agirait donc ici que d'une généralisation de ce principe.

Le problème relevé en (b) fait que la liste de composantes proposées peut très vite se retrouver vide laissant une liste extrêmement longue de résultat à partir horizontalement, et sans faire un clic hors de la boite de dialogue par inadvertance car cela la ferme.

5 Structure composée des sinogrammes

Ce travail se base sur une caractéristique intéressante de l'écriture chinoise : le fait que nombre de ses caractères partagent des parties graphiques qui vont au-delà des traits de base. Il me paraît important d'expliquer, tout du moins d'essayer, la raison de cette particularité.

5.1 Définitions

Puisque nous traiterons d'écritures logographiques et de logogramme, il convient de les définir précisément afin que la notion soit claire dans l'esprit du lecteur qui ne l'a connaîtrait pas tout autant que de dissiper d'éventuels doutes émergents de définitions parfois forts différentes comme il en existe en linguistique.

5.1.1 Logogramme

Pour cela, nous utiliserons la définition suivante du logogramme, fournit par l'encyclopédie Wikipédia :

> « *Un logogramme (du grec ancien λόγος, « parole », ici « mot », et γράμμα, « caractère, lettre ») est un unique graphème notant un lemme (mot) entier et non seulement une partie de ses phonèmes. Dans la majorité des cas, rien n'indique, dans un logogramme, son signifiant (comment il doit être prononcé). En d'autres termes, c'est la plus petite unité significative du langage comme signe unique écrit qui représente un mot complet, indépendamment de la langue.* »

Cette définition me paraît en effet plus pertinente que celle fournit par la version web du dictionnaire Larousse, qui indique :

> « *Dessin correspondant à une notion (logogramme sémantique ou idéogramme) ou à une séquence phonique (logogramme phonétique ou phonogramme).* »

Cette dernière a en effet le défaut de considérer le logogramme comme un dessin, ce qu'il n'est pas : il s'agit d'un caractère textuel et non d'une image (en dehors du fait qu'on pourrait qualifier d'image tout élément perceptible visuellement). Si ce n'était pas le cas, nous ne serions plus dans le domaine de l'écriture, mais dans celui de l'art.

La définition donnée par le Grand Dictionnaire Linguistique & Sciences du langage publié chez Larousse, bien que plus fournie n'en est guère plus convaincante, car non seulement elle tombe dans le même écueil que précédemment, mais en outre elle définit la notion de logogramme comme appartenant au domaine d'études des « *écritures idéogrammatiques* ». L'idéogramme y étant compris comme le constituant principal de ces écritures, ce qu'il n'est pas, et qui est la raison même de l'utilisation d'un autre terme, celui de logogramme.

Elle a en revanche le mérite de signaler l'existence de *déterminatifs* au sein des écritures logographiques, point sur lequel nous aurons l'occasion de revenir.

> « *Dans les descriptions des écritures idéogrammatiques, comme l'écriture hiéroglyphique égyptienne en son état ancien, on appelle* logogramme *le dessin correspondant à une notion (logogramme sémantique ou idéogramme) ou à la suite phonique constituée par un mot (logogramme phonétique ou phonogramme) ; enfin, certains logogrammes (ou déterminatifs) sont utilisés comme signes*

diacritiques permettant de préciser l'interprétation à donner d'un signe pouvant lui-même avoir plusieurs sens. »

5.1.2 Écriture logographique

Nous tiendrons comme écriture logographique « un système d'écriture qui fait majoritairement usage de logogrammes ». Citons à titre d'illustration l'écriture égyptienne hiéroglyphique, l'écriture cunéiforme du sumérien, l'écriture chinoise et l'écriture maya.

Dans la section suivante, nous nous pencherons plus en détails sur l'écriture sumérienne et la problématique des homonymes. Puis nous comparerons les solutions trouvées par les écritures sumérienne et chinoise destiné à résoudre ce problème.

5.1.3 Clef et Composante dans l'écriture chinois

En français, le terme communément appelé pour traduire le terme 部首 (bushu) est « clef » ou « clé ». Par rapport à l'anglais *radical*[6], il ne pose pas de problème d'ambiguïté : il s'agit de la partie d'un caractère qui sert à l'indexation d'un caractère dans un dictionnaire. Elle est souvent définie comme étant « la partie la plus saillante du caractère » (Abel-Rémusat, 1827).

En revanche, l'interprétation de cette partie saillante est parfois problématique, puisqu'elle n'est pas forcément la même entre plusieurs dictionnaires de référence et que la liste des éléments graphiques considérés comme étant des clefs a varié au cours de l'histoire lexicographique de l'écriture chinoise. Aujourd'hui un consensus existe toutefois un consensus autour des 214 radicaux dit Kangxi, du nom de l'empereur ayant ordonné la compilation du dictionnaire ayant fixé cette liste, avec parfois quelques variations.

Le terme de « composante (graphique) » d'un caractère recouvre le même type d'objet que ceux listé par les clefs, plus tous les éléments non clefs (parties moins saillantes) d'un caractère. Ce sont donc les constituants graphiques des caractères, qui existent sans référence à une liste fixe de références servant à l'indexation, comme l'est celle des clefs.

5.2 Homonymes dans les écritures logographiques

5.2.1 Comparaisons entre écritures logographiques

Puisque le chinois n'est pas la seule écriture logographique existante, mais qu'elle est la seule (à ma connaissance) faisant usage d'un très grand nombre de caractères composés et que certains de ces caractères peuvent le résultat d'un nombre élevé (supérieur à deux) de composition, il me parait important de se poser la question du pourquoi, et de tenter d'y apporter une réponse en la comparant à d'autres écritures logographiques connues.

5.2.2 Cas de l'écriture sumérienne

Afin de comprendre sommairement le fonctionnement de l'écriture sumérienne et de son usage des déterminants, voici un extrait traduit en français depuis l'anglais de la thèse de doctorat de Jagersma (2010). La note est également traduite du document originel.

[6] Kordek, 2013. p. 59.

« Dans sa version complètement développée, l'écriture sumérienne est une combinaison d'écriture logographique et phonétique. Il y a essentiellement deux types de signes : les signes de mots (logogrammes) et les signes de sons (phonogrammes). Les logogrammes expriment des mots (c'est à dire des lexèmes). Les translittérations du sumérien standard ne font pas la distinction entre logogramme et phonogramme, mais la présente grammaire transcrit les phonogrammes en italique.

> De nombreux signes ont plus d'un usage. Le signe KA, par exemple, est utilisé comme logogramme pour les mots **ka** « bouche », **zú** « dents », **kiri$_3$** « nez », **du$_{11}$.g** « parler » ou **inim** « mot ». Le même signe est également utilisé comme phonogramme ayant pour valeur ka.[7] Un autre exemple de signe multifonctions est NI. Il est utilisé comme logogramme pour les mots **ì**, **ni**, né et **lí**.
>
> Pour désambiguïser les lectures, certains logogrammes et phonogrammes sont utilisés en tant que signes auxiliaires. Ils sont translitérés en exposant. Les logogrammes auxiliaires sont appelées des déterminatifs. Ils indiquent que le mot précédent ou suivant appartient à une classe sémantique spécifique. Le logogramme **diĝir** « dieu », par exemple, est utilisé comme déterminatif avant les noms de dieux : ^{d}**inanna** « (la déesse) Inanna », ^{d}**en-líl** « (le dieu) Enlil ». […] »

Ainsi en sumérien l'ambiguïté qui peut exister sur le sens d'un signe, et donc sur le mot auquel il renvoi, est levée à l'aide d'un autre caractère antéposé ou postposé. Si la translittération indique à l'aide de la typographie qu'un caractère est utilisé comme déterminatif, ce n'est pas le cas dans le corps du texte d'origine. Le même mécanisme n'existe pas en chinois : aucun signe n'est affecté d'une valeur phonétique nulle de manière à préciser la classe sémantique d'un mot qui suit ou qui précède.

> « L'écriture sumérienne n'a jamais eu un signe pour chaque mot différent. De nombreux mots portent un sens qui ne peut pas être figuré. En outre le nombre de signes différents qu'il aurait été nécessaire pour représenter chaque mot individuel aurait impossible à mémoriser. Ces problèmes sont résolus en leur permettant d'être multivalent : depuis relativement tôt, in signe peut représenter deux mots ou plus. Les logogrammes peuvent acquérir des nouvelles valeurs de deux moyens, qui émanent de la nature logographique même de l'écriture. Un logogramme existant représente toujours un mot, c'est-à-dire une unité phonétique et sémantique. De fait, un logogramme qui représente un mot donné peut acquérir une nouvelle valeur soit par association avec le sens de ce mot soit par association avec sa prononciation. » Jagersma (2010). Traduction personnelle de l'anglais vers le français.

Les mots qui peuvent exprimer le signe KA dans la première citation sont repris par l'auteur pour illustrer le cas d'un signe qui acquiert le sens de nouveaux mots par association avec le sens du mot d'origine (bouche).

5.2.3 La solution chinoise à l'homonymie

En chinois, la notation de plusieurs mots différents à l'aide du même caractère montre que ce mécanisme y est également à l'œuvre (sans avoir recours à l'étymologie, il peut d'ailleurs être difficilement de distinguer s'il s'agit d'homonymes, de polysémie ou dans le cadre d'une traduction vers une autre langue, d'une polysémie qui existe dans la réalité linguistique de la langue cible mais

[7] Dans les translittérations les accents et les nombres souscrits permettent de distinguer les signes : **du** est une valeur du signe DU, **dú** ('**du** deux') une du signe TU and **dù** ('**du** trois') une du signe GAG, … mais à partir de quatre, les valeurs sont indiquées à l'aide d'un chiffre souscrit plutôt que par de accents. La valeur **du$_{11}$** ('**du** onze') appartient au signe KA.

pas dans la langue source). Par exemple en chinois classique 度 lu *dù* est un nom qui désigne une mesure, tandis que lu *dúo* il s'agit du verbe mesurer.

Le second mécanisme décrit est celui de l'association par la forme orale : si un caractère C écrit un mot M_1 de sens S_1 et de prononciation P_1, il peut advenir que les locuteurs s'en servent de manière à écrire un mot M_2 de sens S_2 et de prononciation P_1.

L'association sémantique pose un problème d'ambiguïté ne peut de toute manière pas être levé de manière efficace par un déterminatif sémantique de par sa nature même (mais peut l'être par la syntaxe ou le sens global du reste du message) qui se retrouve pour l'association phonétique. Cette fois, comme les mots appartiennent effectivement à des catégories sémantiques districts, il est possible de lever ou réduire l'ambiguïté à l'aide de la sémantique.

L'originalité de l'écriture chinoise dans la manière de désambiguïser un logogramme réside dans le fait que contrairement au sumérien et à l'égyptien[8] il n'est pas fait usage de déterminatifs dans les phrases ou les énoncés, mais que l'ambiguïté est levée au niveau du logogramme lui-même. Le mécanisme à l'œuvre est le suivant : à partir d'un caractère C désignant plusieurs homonymes (M_1, M_2) appartenant à des champs sémantiques différents, un nouveau caractère C_2 est créé en composant graphiquement le caractère C avec un radical appartenant au champ sémantique de M_2.

De cette manière le caractère originel est soulagé d'un sens, tandis que le nouveau caractère ne désigne qu'un ou plusieurs mots appartenant à un champ sémantique donné. La partie la plus visible est cependant identique entre ces deux caractères et permet d'identifier leur parenté, et le fait qu'il se prononce de la même manière. Des divergences phonétique non parallèles ultérieures à la dérivation rendent plus flou le lien phonétique, voire le faire complétement disparaître.

Chacun des deux caractères résultant du processus pouvant se voir plus tard s'attribuer un nouveau sens par le mécanisme d'association phonétique, il peut exister plusieurs sinogrammes partageant une même composante phonétique et il possible de voir apparaître des caractères qui résultent de plusieurs compositions successives.

5.2.4 Autre originilité de l'écriture chinoise

Une seconde originalité du chinois par rapport aux deux autres écritures logographiques citées (qui n'est pas nécessaire au présent travail mais qu'il est intéressant de relever en cela qu'elle une clef vers la compréhension de son non glissement vers une écriture phonétique) est que l'usage d'un caractère pour sa seule valeur phonétique ou une partie de sa valeur phonétique pour écrire un mot est limité à la transcription de mots étrangers. La méthode *fǎnqiè* (反切)[9] pour indiquer la prononciation d'un caractère montre pourtant qu'il était envisageable pour les chinois d'utiliser leurs logogrammes de façon à transcrire des sons plus petits que la ou les syllabes associées au sinogramme en question.

[8] Qui fait usage de déterminatifs pour les termes écrits en phonétique pour pallier à la non transcription des voyelles, ce qui engendre une ambiguïté (Champollion, 1836), mais pas pour la même raison que le sumérien.
[9] Pulleyblank (1995), p. 5.

5.2.5 Utilisation pour la langue japonaise

En passant au japonais, qui est une langue non tonale contrairement au chinois médiéval[10], le problème des caractères homophones (dans leur prononciation dite sino-japonaise) persiste et est même aggravé par le système phonologique plus réduit du japonais.

5.3 Décompositions informatisées

Un système basé sur la décomposition de caractères chinois suppose que ces décompositions soient disponibles sous une forme utilisable informatiquement. Il existe plusieurs projets ou produits qui fournissent cela.

Un aperçu comparatif des solutions trouvées sera effectué avant d'en détailler le contenu et les spécificités.

5.3.1 Données de décompositions

J'ai recensé 4 fichiers ou base de données existantes :

Kradfile (kanji radical decomposition), existant en plusieurs versions par plusieurs mainteneurs
KanjiVG, par le créateur du dictionnaire Tagaini Jisho
Character Description Language (CDL), de l'Institut Wenlin
Ideographic Description Sequences (IDS), une section du standard Unicode

Tableau synoptique de ces projets

Nom	Format	Nombre d'entrées	Licence[1]
IDS - Kanji Database	Texte brut structuré[2]	81858	GNU Public Licence
KanjiVG[3]	XML	6744 / 11456	CC BY-SA 3.0 Unported
Kradfile (Monash)	Texte brut structuré[2]	6355[4] / 5801[5] (12156)	EDRDG Licence
Kradfile (kanjicafe)	Texte brut structuré[2]	13108	
CDL	XML	93850	Payant, dépend de l'utilisation

1. Une brève explication sur les licences sera présentée en fin de section.
2. Une ligne de texte correspond à une décomposition.
3. Le nombre d'entrées liste les valeurs des fichiers *main* (principaux sinogrammes) et *all* (toutes les données disponibles, ce qui inclus des variantes).
4. Pour la version proposée par l'université de Monash.
5. Pour la version 2 du fichier, publiée par l'université de Monash.

5.4 Unicode et IDS

IDS n'est pas à proprement parler un fichier ou une base de données de décomposition : c'est une méthode décrite au sein d'Unicode pour représenter des caractères chinois absent de ce même standard.

Unicode est un standard informatique destiné à permettre l'échange de texte au niveau mondial, quel que soit l'écriture ou la langue utilisée. Il vise notamment à régler des problèmes

[10] *Ibid*, p. 6.

d'interopérabilité inhérents à l'utilisation de standard incompatibles, parfois au sein d'une même langue, et qui peuvent conduire à une mauvaise interprétation de la part des logiciels de traitement (*mojibake*).

Contrairement aux autres standards et normes d'encodage des caractères, Unicode n'associe pas directement aux caractères une représentation physique (nombre présent en mémoire ou dans un fichier) mais plutôt un « point du code », qui est un nombre indépendant de sa représentation en mémoire. Cette dernière est variable et dépend de l'encodage effectivement utilisé. UTF-8 est un des encodages Unicode le plus utilisé.

Un point du code est dénoté à l'aide d'un nombre hexadécimal de 4 ou 6 digits et précédés de « U+ » et possède également un nom normatif en toute lettre. Par exemple la lettre « a » a pour point du code U+0061 et pour nom « LATIN SMALL LETTER A ».

Bien qu'Unicode assigne des points du code à plus de 75 000 caractères asiatiques, dont la plupart sont des caractères chinois, de par le fait qu'il y a encore aujourd'hui de nouvelles création et qu'il manque des variantes ou des caractères anciens, le chapitre 12 d'Unicode indique une méthode pour exprimer des sinogrammes non standardisé.

5.4.1 IDC

Cette méthode se base sur les Ideographic Description Characters (IDC, caractère de description idéographique). Dans la version 6.0 du standard, ils sont au nombre de 12 et sont placés dans le bloc[11] Unicode U+2FF0 à U+2FFF. Ils représentent la façon dont peuvent être agencés différents composants graphiques d'un caractère.

Ainsi, le caractère U+2FF0 ⿰ permet de représenter dans une séquence IDS un caractère dont les deux composantes sont disposées verticalement l'une à côté de l'autre ; le kanji 和 (wa, hé : paix) par exemple.

Les composantes elles-mêmes sont représentées à l'aide de caractères chinois ou de caractères de radicaux. Unicode défini deux plages de radicaux : les 214 radicaux Kangxi dans le bloc *Kangxi Radicals* qui va de U+2F00 à U+2FDF (les 10 derniers point du code ne sont pas alloués), et un bloc nommé *CJK Radicals Supplement* allant de U+2E80 à U+2EFF et qui contient 13 point de code non affectés. On pourrait ainsi représenter 和 de la façon suivante : (禾 ⿰ 口).

On composer ces caractères de description et ces séquences de façon à former des expressions plus longues qui rendent compte de sinogrammes plus complexes. Ainsi, on pourrait indiquer la décomposition de 程[12] en (禾 ⿰ (口 ⿱ 王)).

5.4.2 Arité des opérateurs et forme préfixe

Parmi les 12 caractères IDC, que nous appellerons désormais « opérateurs », 10 sont d'arité binaire, c'est à dire qu'ils permettent d'assembler deux composantes. Il s'agit de ⿰, ⿱, ⿲, ⿳, ⿴, ⿵, ⿶, ⿷, ⿸, ⿹ et ⿺.

[11] Un bloc Unicode est un intervalle continu de point du code. Chaque bloc est nommé de manière unique et il n'y a aucun chevauchement entre eux. http://en.wikipedia.org/wiki/Unicode_block

[12] (hodo : degré)

Les deux autres sont d'arité ternaire : ils permettent d'assembler trois composantes. Les voici : ⿲ et ⿳. Contrairement aux précédents, ils posent un évident problème de représentation sous forme infixe, puisqu'il faudrait donc les intercaler une fois... entre trois composantes.

En effet, jusqu'ici j'ai présenté mes exemples sous forme infixe et complètement parenthésée : l'opérateur (IDC) est placé entre ses deux opérandes (les caractères ou les clefs), mais dans le standard Unicode les séquences IDS sont formulées en forme préfixe. C'est à dire que les opérateurs viennent avant leurs opérandes.

De cette façon, la séquence (禾 ⿰ 口) s'écrit (⿰ 禾 口) et la décomposition de 程 a pour séquence (⿰ 禾 (⿳ 口 王)). Cela règle le problème des deux opérateurs ternaires, qu'il est ainsi possible d'utiliser sans ambiguïté. Ainsi le caractère 徹 (tetsu, chè : pénétrer) peut se décrire par la séquence (⿲ 彳 育 攵).

Un autre avantage de la forme préfixe par rapport à la forme infixe est la facilité d'analyse syntaxique (*parsing*) de la séquence par un programme informatique.

5.4.3 IDS

Enfin, les séquences IDS telles que décrites dans le standard ne font pas usage de parenthèse ni d'espace, que j'ai utilisées dans mes exemples précédents afin d'en accroitre la lisibilité et la compréhensibilité. Cela n'a aucune conséquence sur leur lisibilité par un programme informatique puisque la forme préfixe n'est pas ambiguë. Elle permet même d'économiser de nombreux octets de stockage, de par sa forme plus réduite.

Ainsi, ⿰禾口 est une séquence IDS valide pour le caractère 程. Jusqu'à la version 6.0 d'Unicode, il existait une limitation sur la taille d'une séquence IDS (16 points du code) destiné en facilité le traitement, mais celle-ci a été supprimée dans la version 6.1.

En outre, comme les exemples précédents ne le montre pas, il peut exister plusieurs séquences IDS valides pour un même caractère. Le standard lui-même fait mention de point et en donne une illustration dans sa figure 8 du chapitre 12, que je reproduis ici.

Figure 12-8. Using the Ideographic Description Characters

Il y a deux raisons à l'existence potentielles de plusieurs séquences IDS pour un même caractère. Premièrement, l'utilisation dans une séquence d'un caractère qui pourrait lui-même être décomposé dans une séquence IDS. C'est illustré par les points ①, ② et ③.

Les points ⑤, ⑥, ⑦ et ⑧ illustrent quant à eux le fait qu'un caractère puisse être décomposé en utilisant des radicaux qui peuvent être sous la forme sous laquelle ils apparaissent effectivement dans le caractère, ou sous leur forme non déformée.

En outre, une troisième raison à la possible existence de plusieurs séquences IDS pour un sinogramme est mise en lumière par ce même exemple : certains caractères ou radicaux sont normalisés sous plusieurs point du code différent.

5.5 Kanji Database Project

Le Kanji Database Project (2015) fourni des séquences IDS. Les données proviennent à l'origine de la base de données de décomposition du projet CHISE.

Le site du projet est en japonais et il est également disponible en anglais, bien que le contenu soit bien plus réduit dans cette langue.

Un fichier de décomposition est disponible sur le site dans l'onglet 字形ＩＤＳデータ (données IDS sur la forme des caractères). Le fichier proprement dit est hébergé sur le site de partage de code source GitHub, ce qui permet à la fois d'économiser la bande passante pour le site du projet mais aussi aux utilisateurs de suivre les modifications apportées au fil du temps à ce fichier de données.

Un second fichier, maintenue par un certain M. Kawabata est listé sur le site, mais le lien renvoie vers une ressource n'étant plus disponible sur internet.

Ce fichier de données liste des caractères à raison d'un par ligne, sous la forme IDS. Certains caractères ne pouvant pas être représentés par l'encodage utilisé (UCS), ils sont présents sous forme d'entités.

5.6 Entités CDP

Il s'agit de codes numériques affectés par le Chinese Document Processing Lab à des composantes qui ne sont pas listées dans Unicode. Cela permet de faire référence à ces caractères en absence d'intégration dans un encodage. De fait, ils sont représentés sous la forme d'une chaîne de caractère particulière et ne sont pas affichés directement par les éditeurs de texte. Il est cependant possible dans certains d'en avoir une représentation graphique (que peut vouloir un humain) grâce au projet Glyph Wiki[13].

5.7 Problèmes des décompositions IDS

5.7.1 Ambiguïté des décompositions

Comme l'indique Kordek (2013) [traduction de l'anglais vers le français] :

> « Il est souvent présumé que la procédure de décomposition des caractères en composantes est intuitive, et de que fait il n'y a pas besoin de la formaliser ou d'écrire des algorithmes pour chacun de ses aspects. Typiquement, seuls les points problématiques (tel que le traitement des variantes des composantes) sont traités avec attention.
>
> La structuration en composantes est parfois ambiguë. En fait, les règles générales doivent laisser une marge d'indétermination qui doit être résolue au niveau des composantes de bases. Le problème peut être illustré par deux des composantes de 醫 : 殳 et 矢. Il y a deux types de critères de décompositions étymologique et structurel. D'un point de vue étymologique, il n'y a aucune raison de décomposer 矢, malgré les particularités structurelles qui suggèrent la présence de deux composantes : 殳 et 𠂉, qui sont toutes deux attestées dans d'autres sinogrammes. La décomposition dans la caractérologie moderne est basée sur des principes structurels. La composante discutée en est un bon exemple. Malheureusement, les choses sont beaucoup plus compliquées que cet exemple. Une redécomposition de 大 est structurellement motivée, mais n'est pas sans controverse. Parmi quatre base de données de décompositions, deux décomposent 大 en 人 et 一 (Kawabata's IDS database and Wenlin 4.1) ; deux les considèrent comme une composante de base (CDP and CHISE) ; et Fan ne décompose 矢 pas du tout. »

Pour compléter la citation, relevons que 矢 (ya, shǐ : flêche) est décomposé dans KanjiVG en tant que 丿 et 天. Dans une application informatique, on doit régler ce problème. Comme notre but est de permettre de rechercher des caractères à partir de composantes identifiable par un humain, il n'est pas nécessaire de sur-décomposer, c'est-à-dire de trop décomposer, des caractères en des composantes auxquelles ne penserai pas un utilisateur sans connaissance approfondie de la structure des caractères.

Il faut donc choisir un fichier qui décompose les caractères avec une idée similaires et corriger les cas qui ne correspondent pas à ce qui est attendu. Cela implique toutefois une grosse quantité à faire manuellement.

[13] http://glyphwiki.org/wiki/Group:CDP%E5%A4%96%E5%AD%97

5.7.2 Points de code multiples

Nous avons vu précédemment que les décompositions IDS pour un caractère peuvent être multiples. Cela est problématique pour l'application visée puisque cela complique le processus de recherche. En effet, si on ne travaille qu'avec une décomposition par caractère, il est possible de manquer des caractères ayant des composants communs pour la simple raison que ces radicaux ne seraient pas représentés par le même point du code Unicode.

Si on ajoute plusieurs décompositions par caractère, on n'est pas assuré de traiter le problème soulevé au paragraphe précédent. Et si l'on tente d'ajouter toutes les décompositions existantes à un caractère, on se heurte à une explosion du nombre de décomposition. Ce qui impacte à la fois le l'espace de stockage nécessaire à stocker ces données, mais la complexité et le temps nécessaire à leur traitement ultérieur.

Il convient donc à la fois de limiter le nombre de décompositions possibles pour un caractère en définissant des conventions et de trouver une représentation efficace des décompositions restantes afin d'en permettre un traitement rapide.

5.8 Licences

Une implémentation effective de la méthode de recherche couverte par ce dossier nécessite des données de décompositions. La tâche de créer une telle base de données étant immense, il faut se baser sur l'existant. Cependant, toutes les bases ne peuvent pas être réutilisées de la même manière : c'est la problématique de la licence.

Certaines de ces licences sont directement ou indirectement tirées de licences pour logiciel. Dans certains cas il est nécessaire de payer la licence à l'organisme qui en possède les droits, dans d'autres cas il est possible de s'en servir gratuitement.

5.8.1 Gnu General Public Licence (GPL)

La GNU est une licence de logiciel dit libre. Elle confère aux utilisateurs d'un logiciel le droit d'avoir accès au code source (les fichiers informatiques nécessaires à sa réalisation) et de réutiliser ce code source pour créer d'autres logiciels, à condition de les diffuser eux-mêmes sous licence GPL. C'est pourquoi on parle de licence « virale ».

Cette licence existe en plusieurs versions, la dernière en date étant la 3. Un logiciel peut être placé sous plusieurs licences libres à condition qu'elles ne soient pas incompatibles entre elles, y compris plusieurs versions de la GPL. Ainsi de nombreux logiciels sont distribués sous licence GPL 2.0 et ultérieures.

Le site du projet Kanji Database ne précise pas la version de la GPL utilisée. Cela peut être problématique car toutes les versions de la GPL ne sont pas compatibles entre elles.

5.8.2 Creative Commons

Les licences Creative Commons permettent aux créateurs de tous types d'œuvres de les mettre à disposition du public en réservant certains droits (utilisation commerciale par exemple) tout en cédant d'autres (par exemple celui de redistribution).

La licence utilisées par le projet KanjiVG impose que le créateur soit cité et qu'il soit fait mention d'éventuelles modifications dans les œuvres dérivées et que ces dernières soient placés sous cette même licence. C'est aussi une licence virale.

6 Algorithme de recherche

L'algorithme suppose l'existence des fonctions[14] définies ci-après. Les sinogrammes et les composantes sont recherchés dans un dictionnaire de taille fini et défini au préalable.

Les kanji et les composantes sont considérés comme des éléments appartenant à deux ensembles distincts, respectivement **H** et **C**, même quand leur représentation graphique est identique. Ainsi 斗 élément de **H** est différent de 斗 élément de **C**. Dans le cas où il pourrait y avoir ambiguïté on les notera respectivement 斗 $_H$ et 斗 $_C$.

6.1 Fonctions nécessaires

6.1.1 Recherche de sinogrammes comprenant une composante c

$r_c(c) \rightarrow \{ K_0, \ldots, K_n \}$
où $c \in$ **C** et $K_i \in$ **H** avec $i \geq 0$ et $i \leq n$.
Exemple : r_c(龠) → { 龠、龡、龢、龤、龕 }

6.1.2 Décomposition d'un sinogramme k en ses composantes

$d(k) \rightarrow \{ c_0, \ldots, c_n \}$
où $K \in$ **H** et $c_i \in$ **C** avec $i \geq 0$ et $i \leq n$.
Exemple : d(開) → { 門、开 }

6.1.3 Existence d'une composante c dans un sinogramme K

$e(K, c) \rightarrow$ K si $d(k) \cap c \neq \emptyset$
$\emptyset$ si $d(k) \cap c = \emptyset$
où $K \in$ **H** et $c \in$ **C**.
Exemples : e(開, 門) → 開 ; e(開, 人) → ∅

6.2 Algorithme formel

L'algorithme prend en entrée une liste de sinogrammes (K_0, …, K_n) nommée K_e et produit une liste de sinogrammes (R_0, …, R_n) en sortie. L'ordre des caractères de la liste peut être modifié par une implémentation de l'algorithme préalablement à son exécution.

La première étape construit l'ensemble des sinogrammes qui contiennent dans leurs composantes une composante appartenant à K_0.

$$T_0 = \bigcup_{c \in d(K_0)} r_c(c)$$

L'étape suivante construit un sous-ensemble T_{n+1} à partir d'un ensemble de sinogrammes T_n et d'un sinogramme K_n (avec $n > 0$) pris dans la liste d'entrée de façon à supprimer de T_n les sinogrammes qui ne comportent pas au moins une composante comprise dans l'ensemble des composantes de K_n (c'est-à-dire $d(K_n)$).

[14] Au sens informatique. Le terme mathématique correct est celui de relation.

$T_{n+1} = \{ x \in T_n \mid \forall k \in T_n, \forall c \in d(k), \exists y \mid \forall c_2 \in d(K_{n+1}) \mid e(k, c_2) \neq \emptyset \}$

Cette étape est répétée pour tous les sinogrammes de la liste d'entrée excepté le premier (K_0).

Puisque l'algorithme itère sur les K_n de la liste d'entrée de taille finie, il est terminable.

6.3 Seconde formulation de l'algorithme

La formulation de l'algorithme exprimée précédemment peut être un peu difficile à saisir. Il me parait donc important d'en fournir une seconde, plus compréhensible.

Cette seconde formulation est ensembliste : il s'agit de produire en résultat l'intersection des ensembles produits par l'union de l'application de la fonction de recherche de caractère à toutes les composantes de ce caractère, pour tous les caractères d'entrée.

$$\bigcap_{k \in K_e} \bigcup_{c \in d(k)} r_c(c)$$

On remarque qu'il n'est pas ici fait usage de la façon e(K, c) d'existence d'une composante dans un caractère. On peut donc produire une implémentation qui s'en passe.

6.4 Description de l'implémentation

6.4.1 Structure du fichier de données IDS

Pour le prototype que j'ai écrit dans le cadre de cette étude, je me base sur les données du Kanji Database Project et du fichier IDS.TXT qu'il met à disposition. J'ai au préalable retiré de ce fichier les lignes qui ne contient pas de description de caractère.

Les entrées dans le fichier IDS ont la structure suivante :

- Point de code Unicode (U+XXXX)
- Un espace
- Le caractère décrit
- Un espace
- Une ou plusieurs séquences IDS, séparées par un espace
- Éventuellement une ou des indications géographiques, placées entre crochets, pour ces séquences IDS

Dans une description IDS il peut être fait usage d'une entité CDP pour les cas où le caractère ou la portion de caractère référencée n'est pas encore encodée dans le standard Unicode.

La plupart du temps, lorsqu'il y a plusieurs descriptions IDS, ces dernières sont suivies d'une indication géographique destinée à indiquer où elles sont applicables. Ce n'est toutefois pas toujours le cas.

Par exemple :

U+4E32　　串　　⿻吕丨⿻中口

U+6971　　榛　　⿰木秦[GTV]　⿰木⿱𠂇夭[JK]

U+697D　　楽　　⿱白木[GJK]　⿱白&CDP-89AE;[T]

Le premier exemple comporte deux descriptions sans indications géographiques, tandis que le second en est pourvu. Le troisième exemple illustre l'usage d'entités CDP dans les descriptions.

6.5 Utilisation du fichier IDS.TXT

Le première remarque à formuler est que la multiplicité des descriptions est problématique, en particulier lorsqu'aucune indication géographique n'est fournie. Car lorsque c'est le cas, si le Japon figure parmi elles, il devient possible de le sélectionner plutôt que les alternatives proposées.

Dans le cas contraire, on a alors soit la possibilité de proposer une implémentation qui fait usage des deux à la fois, ce qui la complique, ou on peut faire un choix arbitraire qui pourrait s'avérer surprenant du point de vue de l'utilisateur.

Comme l'implémentation réalisée avec ce dossier n'est qu'à l'état de prototype, le but est de rester simple pour dégager les problèmes techniques fondamentaux qui pourraient surgir. Il a donc été fait le choix de le conserver et de ne servir que de la première des descriptions proposées pour chaque entrées, en faisait fi des éventuelles indications géographiques.

6.5.1 Description non terminale

On remarque rapidement en observant le fichier que de très nombreuses descriptions ne sont pas terminales, c'est-à-dire que les sinogrammes ou les parties de caractères utilisées peuvent eux-mêmes être décomposées.

Puisque la méthode de recherche proposée dans cette étude se base justement sur des descriptions terminales (qui contiennent en plus les caractères non terminaux) rencontrés, on ne peut pas se contenter de lire et stocker les données du fichier IDS.TXT dans une table de hachage pour espérer implémenter les fonctions r_c et d.

6.6 Représentation sous forme de graph

6.6.1 Arbre de décomposition

Voici l'exemple de l'application de la fonction de décomposition pour le caractère 部.

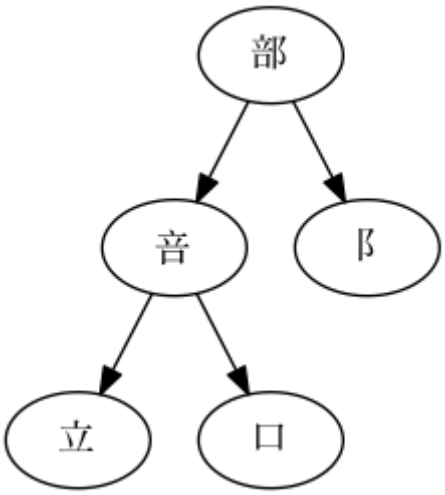


Figure 9 : arbre de composition du caractère 部

Le sinogramme décomposé figure en haut de l'image[15]. Les autres nœuds de l'arbre (音、立、口、阝) constituent le résultat de la fonction d de décomposition.

Ce résultat est produit à partir des lignes suivantes du fichier IDS.TXT :
ligne 17083 : U+90E8 部 ⿰音阝
ligne 1624 : U+5485 音 ⿱立口

6.6.2 Arbre de composition

De manière similaire, il est possible de représenter l'arbre de composition d'un caractère. Ici 部 entre directement dans la composition de 12 sinogrammes. Il entre également via un composé dans un autre caractère qui n'est pas visible sur (en bas à gauche), qui est 鄯. On a un total de 13 caractères dans le résultat de la fonction r_c appliquée à 部.

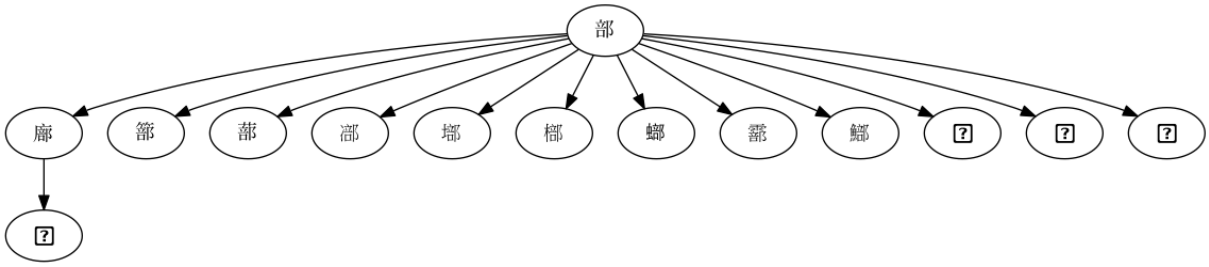


Figure 10 : arbre de composition pour le sinogramme 部

6.6.3 Graph de données

Jusqu'ici les graphs présentés étaient de simples arbres de composition ou de décomposition. Dans les données réelles les références depuis et vers un nœud ne sont pas forcément uniques. L'image suivante montre toutes les compositions du caractère 部 ainsi que les autres clefs et caractères nécessaire à la (dé)composition de ces sinogrammes.

[15] Cette image a été générée semi-automatique à partir de l'exécution de la fonction de décomposition, des véritables données d'entrées et du logiciel Graphviz. Il ne s'agit donc pas d'un exemple fictif.

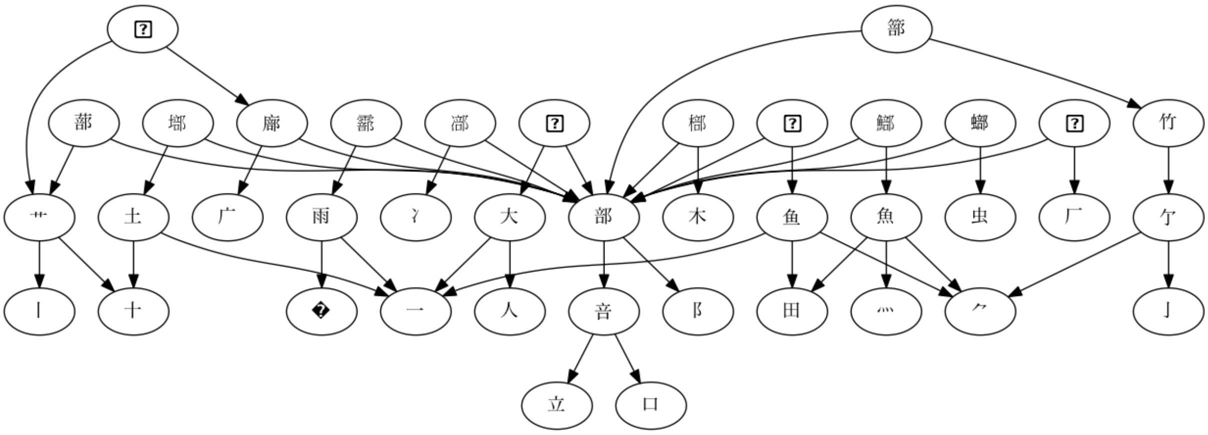


Figure 11 : graph des compositions et décompositions des caractères qui contiennent 部 en composante

Mais ici encore, le graph présenté ne se concentre que sur un caractère. Les données du fichier IDS forment en réalité un immense graph similaire à celui-ci dans lequel des milliers de caractères et de portion de caractères se référencent les uns les autres.

6.7 Implémentation des fonctions r_c et d à partir de graphs

Le graph est une structure de données bien étudiée en informatique pour lequel il existe deux implémentations principales : la liste d'adjacence et la matrice d'adjacence.

Si on dispose d'un graph orienté dont les arcs vont dans le sens nœuds composé -> composantes, l'implémentation de la fonction de décomposition est triviale : à partir du nœud dont on cherche la décomposition, il suffit de suivre récursivement les arcs qui partent de ce nœud et d'ajouter à la liste des résultats les nœuds rencontrés.

Si on dispose d'un graph orienté dont les arcs vont dans le sens nœuds composantes -> composé, l'implémentation de la fonction de recherché de caractères composés est triviale : à partir du nœud dont on cherche les composés, il suffit de suivre récursivement les arcs qui partent de ce nœud et d'ajouter à la liste des résultats les nœuds rencontrés.

On remarque qu'on peut implémenter les deux fonctions de manière identique : seule l'orientation des arcs du graph utilisé en entrée change. On peut donc soit utilisé un graph unique orienté où chaque arc est étiqueté avec sa sémantique (composé -> composantes ou l'opposé) ou au contraire utilisé deux graphs non orienté, chacun ne comportant que les arcs correspondant à une deux des sémantiques en présence.

Comme j'avais à disposition une classe implémentant un graph non orienté, j'ai décidé d'utiliser la solution qui se base sur deux graphs : l'un représente les décompositions, l'autre les compositions. Les fonctions r_c et d sont implémentées à l'aide d'une unique méthode, qui est appliquée au graph idoine.

6.8 Problèmes rencontrés dans l'implémentation

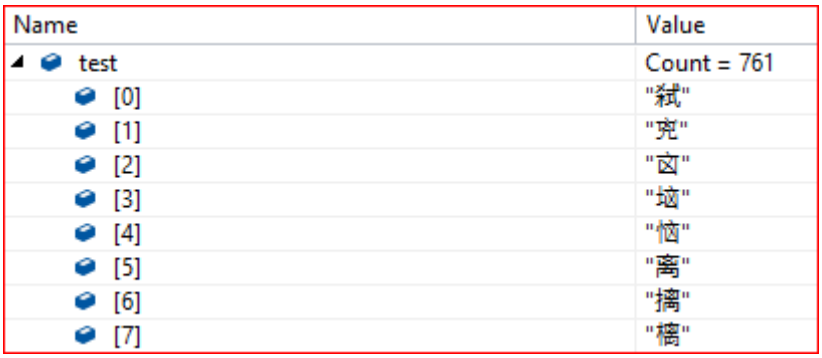
6.8.1 Aperçu du problème

Pour l'instant la méthode implémentée fonctionne dans le sens où les fonctions minimales et l'algorithme ont étés implémentés avec succès. Cependant le nombre de résultats retournés est bien trop élevé.

Prenons pour exemple la situation dans laquelle on veut chercher le caractère 弒 à partir de 殺 et 式. Dans l'idéal on veut que le résultat satisfasse les deux propriétés suivantes :

- Le caractère 弒 est effectivement présent dans la liste des résultats
- La liste des résultats contient peu d'éléments

Malheureusement, on obtient en fait 761 résultats.



La variable « test » est une liste contenant les caractères renvoyés par l'algorithme. On voit à droite qu'elle a une taille de 761 éléments, il y a donc 761 sinogrammes retournés. Les 8 premiers éléments de la liste de résultats sont visibles : 弒 兗 函 垮 恼 离 摛 樆.

Le caractère recherché est bien renvoyé dans les résultats. Sa position en tête de liste provient probablement de la position du caractère dans le fichier de données d'origine et non d'un quelconque classement ayant pour but de deviner l'intention de l'utilisateur. Mais la liste est bien trop longue et ne satisfait pas la seconde propriété recherchée.

6.8.2 Décompositions problématique

Pour comprendre la cause de ce disfonctionnement, observons comment sont décomposés les deux caractères d'entré par la fonction D et les données qui la sous-tendent.

Le sinogramme 弒 est décomposé en 7 composantes : 杀, 乂, 朩, 殳, ?, ? et 又. Les points d'interrogations sont utilisés pour des valeurs qui ne sont pas affichées dans le débuggeur. Ce problème d'affichage peut provenir soit du fait que la police utilisée dans l'outil de développement ne permet pas de les afficher, soit par que les données sont mal lues et interprétées à cause de leur encodage et de celui utilisé en interne par le langage de programmation. Notons que s'il s'agit de ce cas, les résultats ne sont pas affectés tant que les valeurs corrompues sont en bijection avec les valeurs des valeurs qu'elles devraient avoir (non corrompues).

Name	Value
dkorosu	{System.Linq.Enumerable.DistinctIterator<string>}
Non-Public members	
Results View	Expanding the Results View will enumerate the IEnumerable
[0]	"杀"
[1]	"乂"
[2]	"朮"
[3]	"殳"
[4]	"口"
[5]	"口"
[6]	"又"

Figure 13 : ensemble résultat pour la décomposition de 杀

Vu la position de ces points d'interrogations il s'agit vraisemblablement d'une décomposition de 杀 et / ou sa partie supérieur uniquement. On constate qu'on a à la fois dans la décomposition des portions minimales (乂, 朮) et d'autres qui ne le sont pas (殳). Ce n'est pas une erreur car la méthode est effectivement conçue pour pouvoir travailler sur des portions non minimales de caractères. En revanche, l'excès de précision dans la décomposition (qui apparaît également dans d'autres projets de données comme KanjiVG) est préjudiciable au résultat qu'on veut obtenir.

Cela est plus flagrant avec le second caractère de l'exemple : X, décomposé en 弋, ?, ?, 丶 et 工. À nouveau les points d'interrogations indiquent probablement le trait horizontal et le trait courbe qui forment 弋 en plus du point qui a son caractère propre. Ces traits sont ici superflus vis-à-vis de la façon dont est perçu le caractère pour un être humain, mais en plus ils ont un effet délétère sur les résultats car étant des traits très courants ils conduisent à produire un ensemble des kanji contenant une des composantes du kanji d'entrée beaucoup trop grande.

Ce type de problème répété sur plusieurs caractères pose deux problèmes : (1) l'explosion du temps de calcul. Une composante peut se retrouver dans 10 000 caractères. Il faut les retrouver mais aussi chercher la décomposition de ces caractères lors du déroulement de l'algorithme. Ce qui prend beaucoup plus de temps que quand l'ordre de grandeur de cet ensemble est un ou deux fois plus petit. (2) Des ensembles trop grands et trop généralistes en tant résultat. Dans l'idéal l'algorithme présente en sortie un nombre de résultat de l'ordre de grandeur de l'unité. Plus de résultats conduisent à une méthode peu utile.

En résumé le problème provient de la façon dont sont décomposés les caractères dans le fichier IDS.TXT (kanji database project). Il ne peut pas être facilement résolu. Car si on peut corriger quelques centaines de radicaux très utilisés pour corriger du même coup les décompositions des caractères qui s'en servent, c'est en réalité la décomposition de tous les caractères du fichier qu'il faut revoir en adoptant des règles et en faisant des choix propres à pouvoir utiliser les données de la façon qui est envisagée dans cette étude.

6.9 Code source

Le code source du prototype, qui contient les données nécessaires à son exécution, est disponible à l'adresse suivante : https://github.com/titanix/m1_kanji_search.

Le logiciel Visual Studio est nécessaire pour compiler le projet depuis ses sources. Il peut être obtenu gratuitement via ce lien : https://www.visualstudio.com/en-us/products/visual-studio-community-vs.aspx.

6.10 Application pratique

Les écrans tactiles des smartphones et des tablettes permettant amplement de chercher des caractères relativement complexes. Ce n'est pas le cas d'un nouveau type d'appareil qui commence à émerger : les montres connectées, aussi appelées *smartwatch*.

Bien que pourvu d'un écran tactile à l'instar des téléphones intelligents et des tablettes numériques, celui-ci est de taille particulièrement réduit. Ainsi l'écran de l'Apple Watch commercialisée par Apple a une hauteur inférieur à 4 cm et d'environ 3,5 cm. Cela rend le tracé de caractères complexes difficile voire impossible.

Pour autant, il peut être intéressant d'avoir à disposition un dictionnaire de japonais sur une montre connectée. Il faut cependant régler le problème de la recherche de sinogramme complexe afin avoir d'un produit utilisable.

6.10.1 Workflow de recherche d'un kanji complexe sur une montre connectée

L'image suivante illustre les écrans nécessaires à la réalisation d'une fonctionnalité de recherche de caractère chinois composé sur un écran de petite taille.

Liste des écrans : (A) accueil et tracé de caractère, (B) sélection d'un résultat d'OCR et (C) résultats proposés par la méthode de recherche traitée dans cette étude.

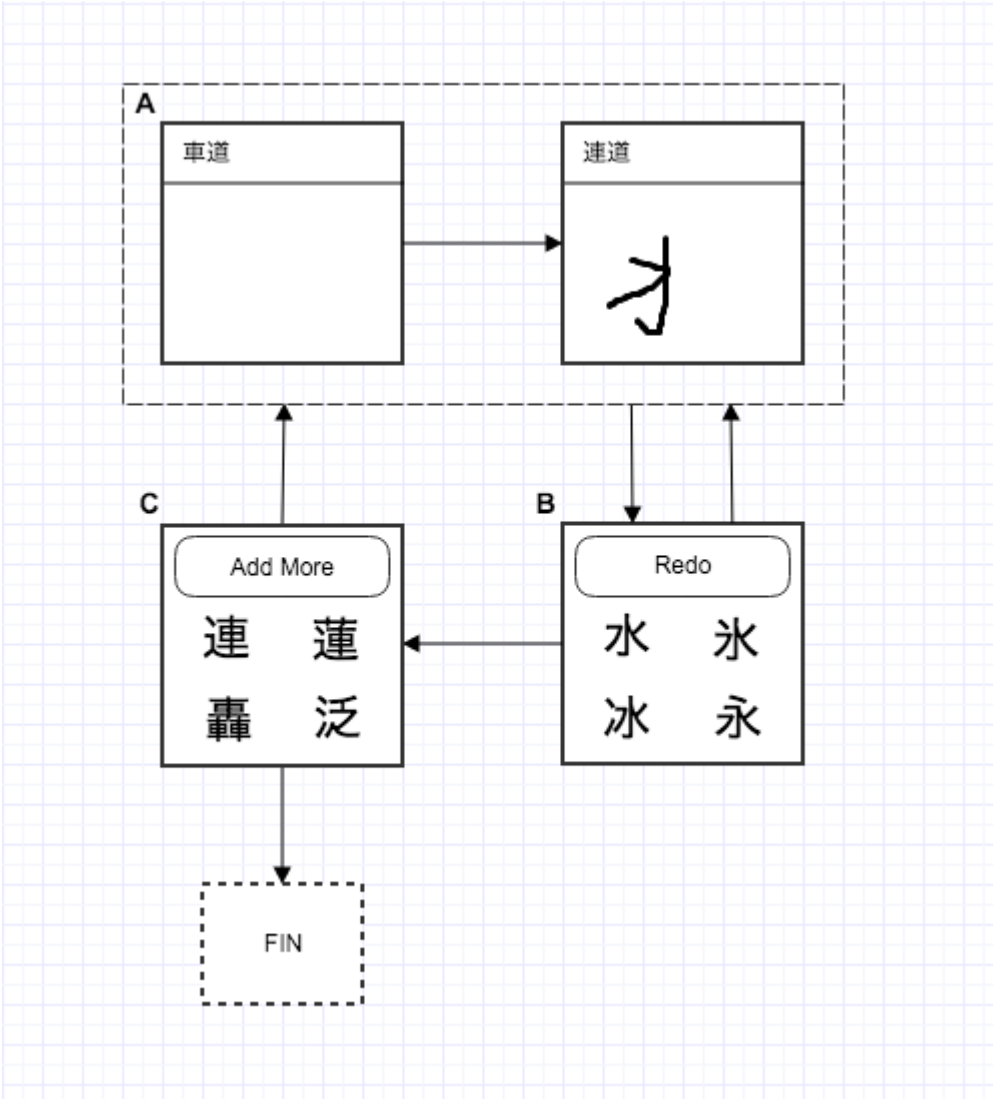


Figure 14 : diagramme de transition entre écrans d'une fonctionnalité de recherche de caractère chinois complexes pour montre connectée

6.10.2 Écran A : tracé

Le premier écran est celui sur lequel démarre l'application. Il comporte deux parties : une partie supérieure qui indique les caractères déjà reconnus et qui sont ajoutés à la liste des caractères d'entrés pour la méthode de recherche. Au début cette zone est vide. Elle est peu à peu remplie grâce à la seconde zone et à l'écran B.

La seconde de l'écran A, située dans sa partie inférieure, permet à l'utilisateur d'écrire un caractère peu complexe sur l'écran tactile de l'appareil. Le tracé sera analysé par une technologie de reconnaissance de caractère (OCR) afin de pouvoir le manipuler sous forme de texte.

6.10.3 Écran B : sélection d'un caractère de recherche

Le résultat de cette phase d'OCR (qui peut déportée sur un téléphone associée à la montre intelligente si besoin est) produit un ensemble de caractère qui est affiché sur l'écran B. L'utilisateur doit choisir un caractère parmi les caractères proposé pour pouvoir passer à la suite. Il peut y avoir

plus de caractères proposés : le défilement vertical permettra à l'utilisateur de voir les autres choix qui s'offrent à lui.

Si le caractère auquel pensait l'utilisateur n'apparaît pas dans la liste, il a la possibilité de cliquer sur le bouton « Redo » présent dans la partie supérieure de l'écran afin de revenir à l'écran A pour pouvoir redessiner le sinogramme.

6.10.4 Écran C : sélection du résultat

S'il trouve le caractère qu'il cherchait, il peut cliquer dessus. Ce caractère est alors ajouté à la liste des sinogrammes utilisés pour la recherche d'un caractère plus complexe et l'écran C lui est présenté. Cet écran contient le résultat de la méthode de recherche pour les caractères présents dans la liste.

Si aucun caractère n'est celui que recherchait l'utilisateur, il peut affiner sa recherche en cliquant sur le bouton « Add More » afin de revenir à l'écran A. Si le caractère qu'il cherchait est présent dans la liste, il peut cliquer dessus. Le processus de recherche prend alors fin, et ce caractère devient disponible pour la suite du traitement envisagée dans le reste de l'application.

7 Conclusion

Nous avons abordé la structure des caractères chinois, une hypothèse concernant son origine et comment nous pourrions en tirer une méthode permettant de rechercher des caractères complexes dans un dictionnaire électronique.

La réalisation d'un prototype a permis de mettre en lumière une difficulté inattendue : la précision des décompositions joue contre les résultats attendus. En l'état les données d'entrées ne permettent pas d'obtenir une réponse acceptable à la méthode de recherche proposée. En effet, comme l'indique Kaiser (1996) dans l'introduction de son article, ces décompositions ne sont pas opérés avec en tête l'intérêt d'un apprenant.

Or c'est ici un point crucial : il faut que l'algorithme puisse produire en sortie des résultats cohérents avec le modèle mental de décomposition d'un apprend. Celui-ci est beaucoup moins précis que la source de données utilisée. Pour avoir précédemment essayé de travailler avec les données de KanjiVG, je sais qu'un problème identique se pose avec cette source-là, on ne peut donc se contenter de changer la source de données utilisée. Il est donc nécessaire de recréer un tel fichier en prenant parti pour des décompositions qui correspondent à ce modèle.

Vu le nombre de caractère mis en jeu c'est un travail de longue haleine. Il est probablement possible d'en automatiser une partie, mais *in fine* une révision humaine de chaque entrée devra être effectuée. Je n'ai d'ailleurs pas eu le temps de fournir avec cette étude un fichier de données qui correspondrait à ces critères, comme j'aurai voulu le faire.

En outre la création d'un tel fichier pourrait être l'occasion d'une réflexion pédagogique à la fois quantitative et qualitative sur les portions de caractères qu'il pourrait être intéressant de prioritiser dans les enseignements du japonais.

Enfin, cette méthode prend son sens dans l'optique d'une utilisation sur écran tactile de taille très réduite. À ce titre il sera intéressant de réaliser un prototype pour montre électronique quand le problème précédent sera résolu et d'intégrer celui-ci à une application de dictionnaire sur smartphone.

De cette façon, on augmente la possibilité d'apprentissage en tout lieu et en tout moment qui est latente à ces appareils électronique, mais attend d'être exploité par les développeurs afin d'atteindre tout son potentiel.

En outre il peut être intéressant d'appliquer ceci à la recherche de Chữ Nôm. En effet, si des outils de reconnaissance tactile existent pour le japonais et le chinois, ce n'est pas le cas pour cette écriture qui n'est plus utilisée et n'est donc pas supporté dans les logiciels. C'est cependant un pan important de la culture vietnamienne, et pour sa sauvegarde il est nécessaire d'atteindre des populations plus jeunes. Rendre accessible la recherche de ces caractères peut donc être intéressante pour un dictionnaire ou toute autre application qui s'en servirai.

8 Références

8.1 Articles

Bottéro, Françoise (2002). *Revisiting the wen and the zi: The Great Chinese Characters Hoax*. Bulletin of the Museum of Far Eastern Antiquities 74, 2002, p. 14-33.
Tiré de crlao.ehess.fr/docannexe/file/1513/bottero.wen_zi.pdf

Sampson, Geoffrey, Chen, Zhiqun (2013). *The Reality of Compound Ideographs*. Journal of Chinese Linguistics, Volume 41, Number 2, p. 255-272.
Tiré http://www.grsampson.net/aroc.pdf

Morioka, Tomohiko (2002). ポスト文字コード時代の文書処理技術に関する展望 (posuto moji kōdo jidai no bunsho shori gijutsu ni kansuru tenbō ; Perspectives concernant les techniques de traitement des documents dans une ère post-encodage), 全国文献・情報センター人文社会科学学術セミナーシリーズ (zenkoku bunken jōhō sentā jinbun shakai kagaku gakujutsu seminārshirīzu ; Référence nationale, Série de séminaires du centre d'information des sciences sociales et humaines), No.12 (2002 年 11 月) p. 59-70.
Tiré de http://www.chise.org/papers/dc2002.pdf

高田　智和 (Takada, Tomokazu) (2006). 「新しい文字」の発生とその要因 (Atarashii moji no hassei to sono yōin ; L'apparition de « nouveaux caractères » et ses causes). 特集 新しい日本語 (Tokushū Atarashii Nihongo ; Édition spéciale Le nouveau japonais) 2006-08. 掲載巻 25, 掲載号 9, 掲載通号 310, 掲載ページ 28〜36.

Kaiser, Stefan (1996). 漢字情報データベース(KID)のデザインの問題点 : 構造について. (kanji jōhō dētabēsu(KID) no dezain no mondaiten : kōzō ni tsuite. The Design of a Kanji Information Database (KID) : some problems concerning kanji structure. Journal of Japanese language education methods 3(1), 22-23, 1996-03-30.
Tiré de http://ci.nii.ac.jp/lognavi?name=nels&lang=en&type=pdf&id=ART0009964410

8.2 Livres

Abel-Rémusat, Jean-Pierre. (1827). *Notice sur l'Encyclopédie japonoise, et sur quelques ouvrages du même genre*. Dans *Notices et extraits des manuscrits de la bibliothèque du Roi et autres bibliothèque. Tome Onzième*. Paris : Imprimerie royale.
Tiré de https://play.google.com/books/reader?id=Z9BI-is6byQC&printsec=frontcover&output=reader&hl=fr&pg=GBS.PR1

Champollion, Jean-François (1836). *Grammaire égyptienne, ou Principes généraux de l'écriture sacrée égyptienne appliquée à la représentation de la langue parlée*. Paris : Firmin Didot Frères. p. 71, 371.
Tiré de gallica.bnf.fr/ark:/12148/bpt6k61025921/f4 ou http://www.lib.uchicago.edu/cgi-bin/eos/eos_page.pl?DPI=100&callnum=PJ1135.C45&object=1

Kordek, Norbert (2013). *On Some Quantitative Aspects of the Componential Structure of Chinese Characters*. Poznań : Wydawnictwo.
Tiré de

http://www.researchgate.net/profile/Norbert_Kordek/publication/259502187_On_Some_Quantitative_Aspects_of_the_Componential_Structure_of_Chinese_Characters/links/02e7e52c56029a4a4b000000.pdf

Pulleyblank, Edwin G. (1995). *Outline of Classical Chinese Grammar*. Vancouver: UBC Press.

Ryjik, Kyril (1983). *L'idiot chinois, initiation élémentaire à la lecture intelligible de caractères chinois*. Paris : Payot. 468 pages.

8.3 Thèses

Jagersma, Abraham Hendrik (2010). *A descriptive grammar of Sumerian*. (Thèse de doctorat, Université de Leyde, Leyde, Pays-Bas). p. 15, 16.
Tiré de http://openaccess.leidenuniv.nl/bitstream/handle/1887/16107/Binnenwerk-jagersma.pdf

8.4 Pages web

8.4.1 Base de données de kanji

Breen, Jim. (2008). RADKFILE/KRADFILE.
Tiré de http://www.csse.monash.edu.au/~jwb/kradinf.html

CHISE. (2012). 漢字構造情報データベース (kanji kōzō jōhō dētabēsu ; Base de données informatisée sur la structure des caractères chinois).
Tiré de http://www.chise.org/ids/

Kanji Datase Project. (2015). 字形ＩＤＳデータ. (jikei IDS dēta ; Données IDS sur la forme des caractères)
Tiré de http://kanji-database.sourceforge.net/ids/ids.html

Wenlin Institute. (2015). Character Description Language (CDL).
Tiré de http://www.wenlin.com/cdl#stat

8.4.2 Dictionnaires

Sturgeon, Donald (éditeur). (2011). Chinese Text Project. 說文解字 (shuōwén jiězì).
Tiré de http://ctext.org/shuo-wen-jie-zi

Éditions Larousse. (2015). Larousse en ligne.
Tiré de http://www.larousse.fr/dictionnaires/francais/

8.4.3 Organisations

Vietnamese Nôm Preservation Foundation (VNPF) 會保存遺產喃. (Hội Bảo Tồn Di Sản Chữ Nôm) (1999-2015). About the VNPF.
Tiré de http://www.nomfoundation.org/About-the-Foundation/About-the-VNPF

Chinese Document Processing Lab. (2013). 漢字構形資料庫. (hànzì gòu xíng zīliào kù)
Tiré de http://cdp.sinica.edu.tw/cdphanzi/

GlyphWiki. (2015). GlyphWiki:どうやって使うのか. (dōyatte tsukaunoka ; Comment faire ?)
Tiré de
http://glyphwiki.org/wiki/GlyphWiki:%E3%81%A9%E3%81%86%E3%82%84%E3%81%A3%E3%81%A6%E4%BD%BF%E3%81%86%E3%81%AE%E3%81%8B

8.4.4 Applications web ou téléchargement

MasanoriSoft. (2008). 人名漢字を説明するサービス（仮）. (namae kanji o setsumeisuru sābisu ; Service d'explication des kanji utilisés dans les noms de famille)
Tiré de http://mpbets.net/kanji/

8.5 Logiciels

Alexandre Courbot. (2008). Tagaini Jisho (version 1.0.3) [Logiciel].
Tiré de http://www.tagaini.net/

Association Ricci. (2010). Le Grand Ricci Numérique (version 1.0) [Logiciel]. Paris : Édition du Cerf.

8.6 Brevets

Eisenhart, Frank J., Pittman, James A., Simard, Patrice Y. (2013). Radical-base classification of East Asian handwriting. Brevet américain n° US8428358 B2. Washington, DC: U.S. Patent and Trademark Office.
Tiré de https://www.google.com/patents/US8428358