Institut National des Langues
et Civilisations Orientales

Département Textes Informatique Multilinguisme (TIM)

Louis Lecailliez

Approches pour une numérisation de qualité d'un dictionnaire vietnamien-français comprenant des caractères Nôm

---

Mémoire pour l'obtention du Master
Traitement Automatique des Langues
Parcours Ingénierie Multilingue

Dirigé par Marie-Anne Moreaux
Novembre 2015

Table des matières

Introduction

Le présent mémoire se propose de réaliser la reconnaissance optique de caractères d'un dictionnaire ancien de vietnamien-français. La particularité de ce dernier est de mélanger plusieurs langues et systèmes d'écriture, dont une écriture indigène calquée sur l'écriture chinoise, qui n'est plus lue que par une centaine de spécialistes, et pour laquelle il n'existe pas d'outils tout faits.

Cela n'empêche pas certains individus de s'y intéresser, qui en ont découvert l'existence sur Internet, mais ces derniers se trouvent vite confrontés à un manque de ressources, en particulier dictionnairiques. Il existe cependant plusieurs dictionnaires vietnamien-français publiés à la fin du XIX^{e} siècle, dont le contenu est libre de droits et dont des numérisations existent. Le format image limite cependant les usages, notamment de recherche dans un dictionnaire, ce qui est pourtant un incontournable de l'apprentissage d'une langue et de son écriture.

Le but de ce travail est de fournir une version textuelle structurée de l'un de ces dictionnaires papier qui puisse être utilisée comme base de données pour réaliser un dictionnaire électronique. En plus des problèmes inhérents aux technologies de reconnaissance de caractères (OCR), il y a des problématiques propres à la coexistence de plusieurs langues et de multiples systèmes d'écriture au sein d'un même document.

En outre, il me semblait intéressant de travailler sur un ouvrage compilé par un ancien professeur de l'Inalco, à l'époque où celui-ci s'appelait l'École spéciale des langues orientales vivantes, et de placer ainsi mon travail dans la continuité de la mission pratique et scientifique de notre établissement, tout en faisant usage de techniques contemporaines pour réactualiser les modalités d'accès à un contenu qui n'a pas d'équivalent dans le monde.

Enfin, puisque c'est par ce mémoire que je dois démonter ma capacité à mettre en pratique des connaissances apprises au cours de ces deux dernières années de formation, ce sujet a soigneusement été choisi pour expliciter ma capacité à faire un usage conjoint de connaissances et compétences diverses au service d'un objectif pratique : linguistique, informatique (programmation fonctionnelle et objet), langagière (vietnamien, chinois) et théorique (langage formel et automate).

Plan

La première partie de ce mémoire est consacrée à une brève présentation des caractéristiques des langues et des écritures asiatiques abordées dans ce document afin que le lecteur, même non orientaliste, puisse comprendre les parties suivantes.

La deuxième partie détaille le corpus et les choix afférents et est suivie d'une analyse de la structure du dictionnaire étudié. À ce propos, différentes méthodes pour tenter d'optimiser les résultats de traitements seront esquissées.

La troisième reprend et donne une vue synthétique des techniques d'améliorations citées précédemment. Enfin la quatrième partie approfondie le cadre théorique de chaque méthode (si besoin est), détaille les choix d'implémentations et les résultats obtenus pour tenter de vérifier si elle est applicable ou non.

Langues et écritures en présence

Le présent mémoire faisant référence à plusieurs langues et plusieurs systèmes d'écritures, il m'est apparu nécessaire de les présenter au préalable afin que le propos ne soit pas gêné par un manque d'information sur ceux-ci pour le lecteur non orientaliste.

Le dictionnaire est directement concerné par la langue française et son écriture, qui ne seront pas présentés ici, ainsi que par la langue vietnamienne et ses deux écritures : l'ancienne écriture logographiques appelée chữ nôm (字喃) basée sur les caractères chinois, et l'écriture utilisée actuellement basée sur l'alphabet latin : le quốc ngữ (國語 langue du pays).

Enfin, le chinois moyen est utilisé pour une des techniques d'amélioration de la reconnaissance de caractères et son écriture, l'écriture chinoise, est également présente dans le dictionnaire et est la base du chữ nôm.

Le chinois moyen

Le chinois moyen[1], plus récent que le chinois ancien qui s'étend du XI^{e} siècle av. J.-C. au III^{e} siècle av. J.-C. (Sagart 1999 : 4), est un état de langue qui irait du VI^{e} au XII^{e} siècles[2] dont la trace est conservée dans certains ouvrages tel que le Qièyùn (切韵). C'est une langue tonale[3], dont de nombreux mots ont été empruntés par des langues, de familles linguistiques différentes, des peuples voisins : japonais, coréen, vietnamien. C'est grâce à ces langues et aux dictionnaires de rimes que des reconstructions ont pu être proposées par des sinologues de renom : Karlgren, Maspéro, Pulleyblank et Baxter en particulier.

En chinois moyen déjà, la langue est monosyllabique. Les syllabes sont analysées traditionnellement sous la forme d'une initiale et d'une rime, rime qui comporte également le ton. La méthode pour expliciter la prononciation d'un caractère à partir de deux caractères chinois, l'un de même initiale, l'autre de même rime, est appelée et fǎnqiè (反切)[4].

Avec cette méthode, le premier caractère donné doit avoir la même initiale que celui analysé et le second la même finale et le même ton. Par exemple en chinois contemporain, 當 qui se lit dāng pourrait être indiqué sous forme de fǎnqiè de la manière suivante : 大方切. 大 qui se lit **d**à et 方 qui se lit f**āng** permettent de noter que 當 se lit dāng. Le caractère 切 sert à indiquer qu'il s'agit d'un fǎnqiè.

On voit que cette analyse traditionnelle est précise dans l'initiale d'un mot, mais ne l'est pas dans sa description de la finale, puisqu'elle ne sépare pas la voyelle de l'éventuelle consonne finale.

[1] Ou chinois médiéval. J'ai choisi d'utiliser un anglicisme, plus court à écrire au vu de sa fréquence dans ce chapitre.
[2] Wikipédia (https://fr.wikipedia.org/w/index.php?title=Chinois_m%C3%A9di%C3%A9val&oldid=114475173)
[3] Qui comprend 4 tons, différents des 4 tons du chinois contemporain.
[4] Pulleyblank 1995 : 5-6.

L'écriture chinoise

Dans les langues chinoises qui s'écrivent avec des caractères chinois, chaque morphème est généralement associé à un caractère particulier[5]. Plusieurs morphèmes différents peuvent être liés à un même caractère.

Pour comprendre la structure très particulière de l'écriture chinoise, qui est utilisée dans une des techniques d'amélioration de l'OCR de ce mémoire, il faut se pencher sur son[6] processus d'évolution. De manière similaire à ce qu'indique Jagersma[7] dans sa thèse *A descriptive grammar of Sumerian* à propos du sumérien, qui est également noté à l'aide d'une écriture logographique, un caractère qui note un mot peut acquérir de nouvelles significations selon deux axes : sémantique et phonétique.

Sur l'axe sémantique, des termes proches du sens que possède le mot d'origine peuvent amener à être écrits par le même caractère. Celui-ci acquiert ainsi la prononciation des signifiés correspondants, qui peuvent être très différents de ce premier mot.

Sur l'axe phonétique, on peut se mettre à utiliser un caractère ayant un signifiant A qui dénote un signifié X pour noter un signifié Y homophone (exact ou partiel) de prononciation A ou A'. Ainsi le sens du caractère doit être déduit du contexte, puisqu'il permet de noter des signifiés très différents. Par exemple 女 (reconstruit *nraʔ en chinois moyen par Baxter et Sagart) qui signifie « femme » a également servi à écrire le pronom personnel de la seconde personne, plus tard noté 汝 (reconstruit *naʔ).

L'exemple met en lumière l'intérêt de créer de nouveaux caractères : pour séparer les différents sens d'un caractère, on adjoint à la graphie existante un autre élément qui en précise la classe sémantique. Il est ainsi plus facile de comprendre quel mot est associé à ce caractère, tout en ayant une indication sur sa prononciation[8], puisque le caractère à l'origine de la dérivation est encore inclus dans le nouveau caractère en tant que composante. Des caractères à la graphie déjà composée peuvent se retrouver plusieurs fois dérivés pour en former de nouveaux, formant ainsi des caractères à la structure facilement identifiable. Dans l'exemple précédent c'est la clef de l'eau (氵, 水) qui a été adjointe au caractère 女 existant.

Ce mécanisme n'a pas seulement été utilisé en Chine pour créer de nouveaux caractères, mais également par les peuples qui ont utilisé les caractères chinois pour noter leur propre langue. Ainsi les japonais ont-ils créées quelques caractères qui leur sont propres, désignés sous le nom de kokuji (国字), tandis que chez les vietnamiens cette façon de procéder a permis la création de très nombreux caractères : les nôm.

---

[5] Les emprunts aux langues étrangères pour lesquels un morphème peut être polysyllabique est un cas particulier évident qui dévie du cas général (Pulleyblank 1995 : 9).

[6] En réalité, un des processus. Pour l'analyse traditionnelle de la formation des caractères, je vous renvoie à Pulleyblank 1995 : 7-8 qui explique en anglais cette analyse, consignée dans le Shuōwén jiězì.

[7] Jagersma 2010 : 15-16.

[8] Notons que ce type de composé phono-sémantique, appelé xíngshēng (形聲), est le plus fréquent dans l'écriture chinoise mais que certains ont été « corrigés » à certaines périodes pour coller à la prononciation du moment ; on retrouve également des corrections de ce type lors de la simplification effectués en République Populaire de Chine au XX^{e} siècle.

L'écriture vietnamienne ancienne : le nôm

La langue vietnamienne partage des traits communs avec le chinois : elle est isolante et monosyllabique et à chaque syllabe correspond un morphème.

Les vietnamiens ont tout d'abord écrit le chinois classique à l'aide de caractères chinois, désignés sous le terme de chữ hán (字漢). Puis est venue l'idée d'écrire la langue locale, également à l'aide de cette écriture. Nguyen (1978)[9] explique que si il est difficile de dater le début de cette utilisation, elle ne peut être plus récente que le VII^{e} ou le VIII^{e} siècle. Ce qui est à un siècle près la période à laquelle les Japonais se sont servis des caractères chinois pour écrire dans leur langue : le plus ancien ouvrage japonais est le Kojiki (古事記) rédigé en l'an 5 de l'ère Wadō (和銅), c'est-à-dire 712 de notre ère.

Il manquait cependant des caractères pour les mots spécifiquement vietnamiens, et c'est pourquoi de nombreux caractères furent créés, selon plusieurs modèles. Le premier a déjà été vu : il s'agit de celui de l'homophonie. Un caractère chinois existant se retrouve utilisé pour écrire un mot vietnamien de prononciation identique ou similaire à celui de la lecture sino-vietnamienne que celui-ci possède déjà[10]. Exemple : 些 lu 'ta' qui signifie « peu de chose » en sino-vietnamien et qui est également utilisé pour figurer 'ta', mot vietnamien signifiant « moi ».

La seconde méthode est d'utiliser un caractère pour son signifié chinois mais d'y associer un signifiant vietnamien. Cette méthode existe également en japonais, c'est pourquoi le sinogramme 山 (montagne) peut se lire 'yama', qui est un mot japonais. Ce caractère peut toutefois être lu en sino-japonais 'san' dans certains composés.

Enfin des caractères de structure composée peuvent être formés en assemblant des caractères ou des portions de caractères. Ces compositions peuvent être de plusieurs natures : élément phonétique + élément sémantique ou deux éléments sémantique (cas rare)[11]. Ces composants peuvent être des caractères chinois ou des caractères nôm.

On voit que le nôm inclut certains caractères chinois qui ne sont pas lus comme des caractères chinois, ainsi que de nombreux caractères forgés pour l'occasion. Ces derniers ne se retrouvent pas en chinois. Ainsi, on peut traiter une partie du problème de l'écriture nôm avec des outils prévus pour le chinois (notamment un modèle entraîné sur des données chinoises) mais que ce n'est pas une approche viable pour la totalité du problème et il est nécessaire d'utiliser ou de créer un outil pour gérer ce sous-problème.

L'écriture vietnamienne moderne

À partir du XVI^{e} siècle plusieurs systèmes de romanisation ont été créés par des missionnaires chrétiens portugais, puis après raffinement l'un d'entre eux a fini par s'imposer sous le nom de quốc

[9] Les exemples donnés dans cette partie sont également tiré de cette source.
[10] Nguyên 1978 : 45.
[11] Nguyên 1978 : 48-49.

ngữ[12]. Alors que l'écriture nôm ne laissait aucun espace entre les caractères, le quốc ngữ sépare chaque syllabe par une espace, sans tenir compte du fait qu'il s'agisse d'un mot ou non[13].

Comme le détaille Huynh (2013 : 13-14), cette écriture ne met pas en bijection phonème et graphème. Il est donc nécessaire d'appliquer un traitement non trivial afin de reconstruire la suite de phonèmes qui forme une syllabe. Dans le cadre de ce mémoire, un transducteur à états finis a été construit pour cette tâche.

[12] Bonet 1899 : v.

[13] C'est pourquoi la segmentation, qui est un problème important car nécessaire à traiter avant toute autre application informatisée s'applique également au vietnamien, au même titre qu'au chinois et au japonais.

Corpus et outils de travail

Corpus de travail

Le site chunom.org, qui expose une petite base de données de caractères et de vocabulaire propose également à ses visiteurs les deux dictionnaires suivants.

Le dictionnaire annamite-français (langue officielle et vulgaire) de Jean Bonnet (大南國音字彙合解大法國音 Đại Nam quốc âm tự vị hợp giải Đại Pháp quốc âm), publiés en deux volumes en 1899.

Le dictionnaire annamite-français (大越國音字漢字法釋集成 Đại Việt quốc âm Hán tự Pháp thích tập thành) deuxième édition de J. F. M. Génibrel.

J'ai choisi de travail sur l'ouvrage de Bonnet, un peu plus aéré graphiquement, dont les scans sont un peu plus lisibles. En outre les expériences seront le plus souvent réalisées avec les images en couleur trouvées sur archive.org plutôt que sur les scans en noir & blanc qui ont vraisemblablement fait l'objet d'un post traitement. Ce corpus sera appelé ARC dans la suite du mémoire. Afin de voir si l'utilisation de la couleur permet d'obtenir de meilleurs résultats, je mènerai également certaines expériences avec la version en noir & blanc sur la version trouvée sur le site de la bibliothèque nationale de France, appelée ici corpus BNF.

Contenu du dictionnaire

En réalité le dictionnaire étudié contient deux volumes. Le corpus de travail n'est composé que du premier volume.

L'archive utilisée est composée de 490 fichiers image encodés en JPEG 2000 nommés sous la forme dictionnaireanna01bone_XXXX.jp2 où XXXX est un nombre décimal allant de 0000 à 0489. Puisque les couvertures sont numérisées, il y a un peu plus de fichiers que de pages de contenu. Dans la suite de cette étude, si les numéros de « pages » sont indiqués à l'aide de 4 chiffres, il s'agit d'une référence à un nom de fichier. Si la page est indiquée à l'aide de moins de chiffres, il s'agit de la numérotation interne qu'utilise le dictionnaire.

L'ouvrage contient plusieurs pages blanches au début et deux fois sa page de titre. La préface commence au numéro 0015 et s'étend jusqu'au 0018. S'en suit jusqu'à la page 0039 une explication dite grammaticale, qui couvre en fait brièvement la phonétique, l'écriture et les dialectes du vietnamien en plus d'indications proprement grammaticales. Cette introduction, même si elle ne serait être suffisante et doit être complétée par la lecture de travaux plus complet et plus récent, est importante à lire pour qui n'a aucune notion à propos de la langue vietnamienne, comme c'était mon cas avant de commencer à travailler sur ce projet.

À partir de 0041 jusqu'à 0480 s'étendent les pages qui contiennent les entrées et les définitions du dictionnaire. Par rapport aux autres pages, la page 0041 contient en outre le titre abrégé de l'ouvrage, qu'on ne retrouve plus sur les pages suivantes.

De 0481 à 0489 on retrouve principalement des pages blanches, ainsi qu'une page de publicité pour les autres publications de l'éditeur en 0482.

Reconnaissance optique de caractères (OCR)

La reconnaissance optique de caractères, appelées en anglais *optical character recognition*, abrégée en OCR, est la conversion d'images contenant du texte (imprimé ou manuscrit) en texte électronique.

Cette conversion est intéressante pour de nombreuses applications pratiques car il est plus commode de manipuler du texte sous format textuel, que sous forme imagée où le traitement nécessite d'utiliser des algorithmes manipulant des pixels.

Pour le projet en cours, l'objectif est de pouvoir récupérer le contenu du dictionnaire sous une forme textuelle structurée nécessitant le minimum de travail de relecture humaine afin de pouvoir s'en servir comme source de données pour des applications de haut niveau tel qu'un dictionnaire électronique ou un outil de conversion de quốc ngữ vers chữ nôm par exemple.

Prétraitements

Un logiciel d'OCR applique généralement plusieurs phases de prétraitement avant de tenter la reconnaissance proprement dite du texte contenu dans une image. Wikipédia[14] liste plusieurs de ces prétraitements, dans l'ordre suivant :

- légère rotation de l'image
- suppression du bruit
- conversion en noir et blanc ou niveaux de gris (1)
- analyse de l'agencement du contenu (*layout analysis*)
- détection des lignes et des mots
- reconnaissance de l'écriture (2)
- isolation des caractères
- normalisation du ratio et de l'échelle

Deux remarques sont à faire concernant ces prétraitements, qui existent ou non selon les implémentations. Premièrement (1) une version du corpus de travail existe en noir et blanc : c'est la version mise en ligne par Gallica, la plateforme en ligne de la Bibliothèque Nationale de France. Mais la version en couleurs étant plus facile à lire pour un humain, j'ai préféré travailler en priorité sur cette dernière.

Deuxièmement (2), la reconnaissance de l'écriture suit des heuristiques propres à chaque langues et systèmes d'écriture. Les résultats peuvent être surprenants[15] si on tente de reconnaître des caractères chinois disposés en carrés : l'ordre d'écriture traditionnel est en effet des colonnes, lues de droite à gauche. Dans le logiciel utilisé (cf. *infra*), on peut forcer un ordre de lecture.

Un exemple de résultat d'analyse de l'agencement du contenu est disponible dans l'annexe A, document 5. Le dessin des rectangles a été réalisé grâce au programme présenté dans l'annexe B, document 2.

[14] Wikipédia (https://en.wikipedia.org/w/index.php?title=Optical_character_recognition&oldid=688253503)
[15] Ce qui n'a pas manqué d'arriver une fois.

Reconnaissance et post-traitement

La reconnaissance elle-même est décrite par Wikipédia[16] de cette façon:

> « There are two basic types of core OCR algorithm, which may produce a ranked list of candidate characters.
>
> Matrix matching involves comparing an image to a stored glyph on a pixel-by-pixel basis; it is also known as "pattern matching", "pattern recognition", or "image correlation". This relies on the input glyph being correctly isolated from the rest of the image, and on the stored glyph being in a similar font and at the same scale. This technique works best with typewritten text and does not work well when new fonts are encountered. This is the technique the early physical photocell-based OCR implemented, rather directly.
>
> Feature extraction decomposes glyphs into "features" like lines, closed loops, line direction, and line intersections. These are compared with an abstract vector-like representation of a character, which might reduce to one or more glyph prototypes. General techniques of feature detection in computer vision are applicable to this type of OCR, which is commonly seen in "intelligent" handwriting recognition and indeed most modern OCR software. Nearest neighbour classifiers such as the k-nearest neighbors algorithm are used to compare image features with stored glyph features and choose the nearest match.
>
> Software such as Cuneiform and Tesseract use a two-pass approach to character recognition. The second pass is known as "adaptive recognition" and uses the letter shapes recognized with high confidence on the first pass to recognize better the remaining letters on the second pass. This is advantageous for unusual fonts or low-quality scans where the font is distorted (e.g. blurred or faded). »

Enfin, toujours selon la même source, il y a une phase de post-traitement destinée à améliorer la qualité de la sortie. Celle-ci est généralement réalisée à l'aide d'un dictionnaire. Le logiciel Tesseract se sert d'un dictionnaire lors de sa phase de segmentation.

Principaux logiciels d'OCR disponibles

Avant de s'intéresser aux solutions logicielles existantes utilisables pour ce projet, il convient de poser des critères de sélection qui permettront de limiter en amont la liste de ces solutions.

Les critères minimums pour qu'un outil puisse être considéré pour ce projet sont les suivantes :

- Gratuit
- Entrainable
- Supporte l'OCR du chinois

La gratuité est imposée pour deux raisons : le budget cette étude est de zéro, d'autre part parce qu'il serait inconvenant pour la facilité de reproductibilité des résultats d'imposer des frais si on peut s'en passer.

L'outil doit de plus être entraînable : on sait d'avance qu'une des écritures du dictionnaire n'a pas de support dans les logiciels grand public. Il faut donc un moyen de pouvoir l'utiliser avec de nouvelles données qu'il ne supporte pas à la base.

[16] Wikipédia (https://en.wikipedia.org/w/index.php?title=Optical_character_recognition&oldid=688253503#Character_recognition)

Enfin, il faut que l'outil supporte la reconnaissance du chinois. De nombreux outils logiciels sont développés par des entreprises, des laboratoires et plus généralement des développeurs occidentaux. De ce fait, beaucoup ne supportent qu'un nombre limité de langues, le plus souvent toutes écrites à l'aide de lettres latines.

Si le vietnamien écrit en quốc ngữ peut probablement être entraîné pour un moteur supportant une langue écrite en caractères romains, la structure très différente de l'écriture chinoise suppose que le moteur ait été conçu en amont pour pouvoir la traiter.

Dans le cas extrême où aucun outil ne répondrait à ces critères, on pourrait séparer le problème de la reconnaissance du chinois et éventuellement utiliser un outil distinct pour ce cas-là.

En dernier lieu, un logiciel open source sera privilégié par rapport à un outil gratuit mais dont les sources ne sont pas disponibles si la qualité des résultats produits par ces logiciels est à peu près équivalente. Cela offre plus de flexibilité au projet.

Choix d'un logiciel

OCROPY

Ocropy est un projet open-source écrit en Python sponsorisé par Google[17] qui utilisait originellement Tesseract comme moteur de reconnaissance mais qui a ensuite été pourvu du sien.

J'avais l'intention de tester ce projet, malheureusement je n'ai pas réussi à l'installer sur ma machine. En effet, il se base sur Python et sur certaines bibliothèques écrites dans ce langage, qui nécessitent elles-mêmes des gestionnaires de packages qui n'ont pas pu être installés.

Tesseract

Tesseract est écrit en C et en C++ et est disponible sous licence Apache v 2.0. C'est à l'origine un logiciel propriétaire développé par la société Hewlett-Packard à partir de 1985. Le code a été publié en open source en 2005 et son développement est désormais sponsorisé par Google[18].

C'est le logiciel dont le nom revient le plus souvent lorsqu'on cherche dans un moteur de recherche généraliste un logiciel d'OCR. Il est gratuit, open-source, entrainable et supporte le chinois. Il répond donc à tous les critères de sélection posés précédemment. En outre, le fait qu'il soit connu fait qu'il sera probablement possible de trouver une solution sur le web à un problème, par rapport à un outil peu connu, peu utilisé et mal documenté.

Ses 30 ans d'utilisation de vie en production démontrent en outre qu'il est de qualité industriel. Il est de plus multiplateforme. C'est pour toutes ces raisons que c'est le logiciel qui a été choisi pour la suite du projet.

Pile logicielle utilisée

Pour mener les tests sur les outils existants, j'ai utilisé plusieurs versions de Tesseract sur plusieurs systèmes d'exploitation. Ces versions sont : *Tesseract Open Source OCR Engine v3.02.02 with Leptonica* (distribué via MacPorts) et Tesseract pour OS X compilé depuis ses sources[19].

[17] Wikipédia (https://en.wikipedia.org/w/index.php?title=OCRopus&oldid=688381959)
[18] Wikipédia (https://en.wikipedia.org/w/index.php?title=Tesseract_(software)&oldid=673594725)

Comme les fichiers de données nécessaires pour l'analyse ne sont pas forcément les mêmes pour toutes les distributions du logiciel, il peut y avoir des différences sensibles dans les résultats obtenus.

Expériences de base

Une fois l'outil sélectionné, la première chose à vérifier est comment il s'utilise et comment il se comporte avec son paramétrage de base sur le corpus. J'ai donc mené les tests suivants : l'OCR d'une page d'un dictionnaire avec chacune des langues en présence, puis pour certains couples de langues.

Pour ces tests préliminaires, c'est la page 22 (fichier 0062) qui est utilisée.

OCR en français

L'analyse en langue française d'un fichier se fait avec l'aide de la commande suivante :

```
tesseract <image> <output> -l fra
```

Le dernier argument 'fra' peut être remplacé pour utiliser une des autres langues installées avec Tesseract. Lors de l'utilisation de cette commande, un fichier texte en utf-8 est généré en sortie.

L'OCR d'un texte en français donne de très bons résultats sur le français : l'écrasante majorité du texte est analysée correctement.

Constat sur le résultat

Le document 1 de l'annexe A présente le résultat annoté (ajout de couleurs) du résultat de l'OCR de cette page. Les parties en français correctement reconnues (mots, parties de mots et typographie) sont indiquées en vert. Les mots mal reconnus sont colorés en rouge. Enfin les mots reconnus comportant uniquement une faute d'accentuation sont écrits en orange.

On remarque qu'il y a 20 chaînes de caractères fautives réparties de la manière suivante : 16 dans la classe rouge et 4 dans la classe orange. En outre, certaines définitions sont totalement reconnues sans aucune erreur.

Autre fait notable, l'analyse de l'arrangement de la page est correctement réalisée et est prise en compte dans la sortie de l'OCR : la sortie suit la logique de lecture en colonne. Il n'y a donc pas à prétraiter les documents pour séparer et analyser séparément ces deux colonnes car c'est correctement gérer par le logiciel.

Erreurs d'accentuation

Les erreurs liées aux accents sont peu nombreuses. Il y en a 4 dans l'exemple.

Forme de sortie	Forme d'origine	Corrigible par dictionnaire ?
Mais	Maïs	Non
Proteger	Protéger	Oui
Forme	Formé	Non
pomme	pommé	Non

On constate malheureusement que ces erreurs ne sont ni facilement détectables ni facilement

[19] GitHub (https://github.com/tesseract-ocr/tesseract)

corrigeables, puisque la forme de sortie est présente pour ¾ des cas dans un dictionnaire, tandis que la forme d'origine a elle peu de chance de se trouver dans un dictionnaire, puisqu'il s'agit de formes fléchies dans 2 des 3 cas non corrigeables.

Reconnaissance du vietnamien

En utilisant les données d'apprentissage préparées pour le français, les portions de texte en vietnamien ne sont logiquement que très rarement correctement transcrites, les exceptions étant les termes qui peuvent être écrits avec le système d'écriture utilisé pour le français, c'est-à-dire ne comportant pas les symboles diacritiques ou les lettres propres au vietnamien.

Gestion des caractères chinois

La gestion des caractères chinois dans la sortie française est une problématique importante : puisque le système d'écriture est totalement différent de celui du français, on ne peut même pas espérer une ressemblance approximative de la sortie comme avec le vietnamien. Pourtant, c'est une composante importante des données qu'il faut pouvoir traiter.

Les caractères chinois sont rendus par de courtes sections de texte discontinues, qui contiennent souvent des lettres accentuées. Parfois, et c'est plus problématique, il n'y a tout simplement aucune autre sortie que quelques espaces à la place de là où il devrait y avoir un caractère chinois.

Le symbole petit 〇 qui sert de symbole de répétition au caractère chinois de l'entrée est rendu de manière consistante par o ou O, le plus souvent isolé, mais parfois accolé à des caractères « poubelles » qui sont des hán nôm ou des chữ nôm dans le texte d'origine. C'est intéressant car si on connait le caractère qui correspond à l'entrée, on peut recomposer facilement une partie du composé qui est défini. Ce symbole est facilement identifiable et la précision de résolution à sa valeur réelle est égale à celle qu'on peut obtenir pour la reconnaissance du caractère de l'entrée en cours.

L'étoile (*) qui indique qu'un caractère provient du chinois peut être reconnu par le logiciel d'OCR (Tesseract) mais cela arrive peut souvent. On ne peut donc pas compter sur la sortie textuelle d'une passe d'OCR en français pour recueillir cette information de manière fiable. En outre, la précision de cette annotation dans le dictionnaire d'origine ne me paraît pas cohérente avec mes connaissances dans cette langue. Nguyen (1978) nous indique en effet que ce dictionnaire est « à manier avec précaution, des erreurs s'y sont glissées ». Cette classe d'erreurs[20] me parait assez fréquente, je ne peux du reste pas juger de celles qui concernent le nôm et le vietnamien.

La typographie

Typographie : la typographie est très bien rendue. La différence entre tiret long et tiret cadratin est souvent maintenue (même si certains tirets sont redoublés). Les virgules après les termes en vietnamiens sont systématiquement présentes, ce qui est bénéfique pour délimiter les différentes zones de textes.

[20] J'ai relevé une erreur de français (un 'le' à la place d'un 'la') et une erreur à propos de la description de la structure d'un sinogramme.

OCR en vietnamien et en chinois

Vietnamien

La sortie de l'OCR en utilisant le vietnamien est exécrable et n'est pas une base de travail utilisable pour la suite.

On est en présence de deux problèmes. Le premier étant que le texte français est extrêmement mal reconnu. Au vu de la proximité des deux alphabets c'est à première vue surprenant. Mais ça l'est moins quand on se penche sur le fonctionnement de Tesseract : le logiciel fait non seulement de l'analyse de mise en page, mais ce sert également de données linguistiques pour tenter de produire la meilleure sortie possible en prenant en compte des données linguistiques situées au niveau du mot. Les mots vietnamiens ayant une structure très différente de ceux du français, on comprend que le résultat généré soit confus. La sortie est fournie dans le document 2 de l'annexe A.

Ce qui est moins compréhensible cependant, c'est que les mots vietnamiens écrits en quốc ngữ soient si mal reconnus.

Pour le décompte des erreurs du vietnamien deux classes dont étaient utilisées : correct et incorrect. Correct est affecté aux syllabes vietnamiennes correctement retranscrites. Incorrect est affecté aux termes qui comportent au moins une erreur, y compris d'accents.

Sur les 84 syllabes écrites en quốc ngữ que comprend la page 35 sont correctes et 49 sont fautives. Autrement dit plus de la moitié des syllabes à reconnaître ne le sont pas : c'est clairement un mauvais résultat. Le décompte est fourni dans le document 3 de l'annexe A.

Voici la distribution des syllabes et des classes de résultats ventilée selon leur position.

	Total	**Correct**	**Incorrect**
Syllabe de tête d'entrée[21]	10	6	4
Syllabe dans un composé	71	29	42
Syllabe dans une explication	3	0	3

Même si l'échantillon est trop réduit pour avoir une grande confiance en cette déduction, on remarque que la plus grande taille des syllabes en tête d'entrée n'impacte pas de manière significative leur reconnaissance : on reste aux alentours d'une moitié d'erreur.

Chinois

L'utilisation du modèle de langue chinoise (écriture en caractères chinois traditionnels) donne une sortie composée de caractères chinois, comme attendu, mais également de caractères latins.

Un test a été réalisé avec la page 197 (0237), qui a pour particularité de comporter quasiment exclusivement des caractères chinois comme entrées principales, à l'exception du caractère qui note le morphème duồng. Le résultat est consultable dans le document 4 de l'annexe A.

Le résultat est de mauvaise qualité : des parties en français sont reconnaissables mais comportent de nombreuses erreurs. Les caractères chinois prennent dans la sortie bien plus de place qu'ils n'en ont

[21] C'est-à-dire la syllabe qui précède la définition principale.

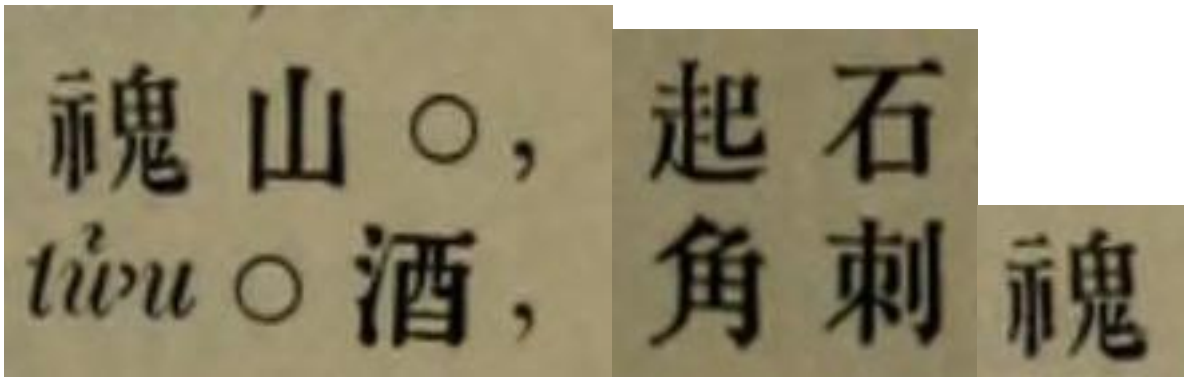
dans le texte d'origine. En outre, les sinogrammes du texte sont mal reconnus. Parmi les 8 caractères des entrées figurantes sur la page, seul 揚 est correctement reconnu.

La raison de ces mauvais résultats est probablement que la partie dédiée à la segmentation est perturbée par la présence de plusieurs écritures sur les mêmes lignes. La largeur d'un caractère chinois étant plus grande que celle d'une lettre latine, une erreur d'heuristique peut mener à de mauvaises segmentations. En outre, pour chaque caractère ou suite de caractères le logiciel doit faire un choix binaire entre deux écritures.

Une solution évidente pour pallier au problème est de ne donner que du chinois en entrée du programme d'OCR, de cette manière toute ambiguïté liée à la nature de l'écriture est levée, la largeur des caractères à analyser est constante sur une ligne de texte, ce qui devrait permettre d'améliorer la reconnaissance du texte en chinois. Il faut néanmoins tester pour voir si cela se vérifie en pratique.

Analyse de fragments isolés en chinois

Pour vérifier si la reconnaissance est améliorée en utilisant seulement du texte chinois en entrée, j'ai réalisé un test avec les trois images suivantes :



La première comporte du chinois, de la ponctuation simple qui existe en chinois, un symbole de répétition qui peut être approximé par le caractère chinois ○ et une syllabe en vietnamien.

La seconde ne comporte que des caractères chinois.

La troisième ne comporte qu'un caractère chinois isolé, présent dans la première image.

Les sorties respectives d'OCR sont les suivantes :

葷鬼 山 O,
祀飆蓬， 互鳳 O 酒蠹

石刺
起角

抨

On remarque plusieurs problèmes avec ces sorties. Tout d'abord, le nombre de caractère en sortie n'est pas le bon même quand le texte n'est composé que de chinois : c'est ce qu'on observe avec la

première ligne de la première image, dont la fin (山 O,) est bien reconnue, mais dont le premier sinogramme est considéré de manière erronée comme un ensemble de deux caractères. Dans cette ensemble, on constate que figure sous forme d'un sinogramme une partie du caractère (鬼) qui est correctement analysée.

Dans la seconde ligne, on voit que la présence de vietnamien se traduit par un grand nombre de caractères chinois en sortie, comme on l'avait déjà remarqué précédemment. De manière plus étonnante, la virgule n'est pas reconnue et est remplacée par un caractère complexe (鼗) . C'est d'autant plus surprenant que la virgule était reconnue précédemment, que le caractère qui la précède est reconnu correctement et que 酒鼗 n'est pas un mot chinois[22].

La seconde image montre que le logiciel peut dans certains cas reconnaître l'intégralité d'un texte en chinois et produire une sortie sans erreur. Ici les caractères n'apparaissent pas dans le bon ordre car c'est l'ordre d'écriture traditionnel qui a été inféré en l'absence d'indication passée au programme.

Enfin, le texte d'origine fait usage du caractère 起[23], qui est une variante graphique (異體字) du caractère 起. En l'absence d'un encodage propre de ce caractère et parce que leur signification est équivalente, on ne peut pas considérer cela comme une erreur.

La dernière image pose une question qui fait écho à la première : puisqu'une partie du caractère était correctement reconnue, mais que le résultat était donné sous la forme de deux caractères, le sinogramme pris en isolation serait-il reconnu correctement ? La réponse est malheureusement négative.

Conclusion

La reconnaissance en français est très bonne et peut directement servir de base de travail. La reconnaissance du vietnamien est en revanche à l'opposée. Il sera probablement nécessaire d'entraîner un nouveau modèle de langue et voir si cela améliore la situation. Quant à la reconnaissance du chinois, qui est par nature plus compliquée, les résultats ne sont pas totalement désespérants, mais il semble nécessaire d'isoler au maximum le texte chinois de façon à ne pas perturber le processus d'analyse, qui est visiblement sensible à cela.

Analyse multilingue

Tesseract permet d'analyser une image qui contient du texte en plusieurs langues en lui donnant en argument les langues susceptibles d'apparaître.

J'ai ainsi testé deux configurations, qui sont des paires de langues, qui pourraient être utiles pour ce travail : français & vietnamien ainsi que français & chinois.

Dans les faits, les résultats ne sont pas bons.

[22] On trouve cependant des collocations de 酒 et 鼗 en cherchant sur Google. Peut-être que le dictionnaire utilisé par Tesseract se base en partie sur de telles données.
[23] Image extraite de Glyph Wiki : http://glyphwiki.org/wiki/u8d77-t

Dans le cas du français et du vietnamien on a un double problème : non seulement les termes en vietnamiens (quốc ngữ) sont mal reconnus, mais en plus puisque le moteur doit désormais faire un choix entre ces deux langues pour chaque terme rencontré, il ajoute donc des erreurs sur la reconnaissance de certains mots français qu'il n'y avait pas quand cette seule langue était attendue par le logiciel.

Contenu du dictionnaire

Agencement d'une page

Les pages de définition du dictionnaire suivent toutes le même arrangement : en haut, centré se trouve le numéro de page, entouré de fioritures typographiques. Ces figures ne seront vraisemblablement pas reconnues par l'OCR, puisqu'il ne s'agit justement pas de caractère. On peut supposer qu'ils laisseront une trace distincte dans la sortie et que le nombre qu'elles entourent sera le plus souvent reconnu correctement.

Les définitions sont ensuite partagées entre deux colonnes, celle de gauche contenant les entrées précédents et celles de droite. Ces deux parties sont séparées par un trait vertical.

Il peut en outre y avoir une lettre centrée sur la page, dans le cas où la page contient le passage à une autre lettre de l'alphabet. Ce passage ne se produit pas forcément en début de page, comme on le constate à la page 9 (0049) lors du passage aux entrées commençant par la lettre B.

Structure des entrées

Puisqu'il s'agit d'un dictionnaire, les entrées sont structurées : elles suivent toutes un format régulier. C'est une propriété très intéressante car cela permet d'avoir des connaissances *a priori* sur la forme du contenu, qui peuvent servir à détecter des erreurs. La succession des entrées elles-mêmes n'est pas aléatoire : elle suit l'ordre l'alphabétique. Ces deux propriétés sont le fondement de cette étude : ce sont elles qui permettent d'espérer qu'on puisse détecter, et pourquoi pas corriger, certaines des erreurs produites en sortie de l'application d'un outil d'OCR.

Vue d'ensemble

Les entrées suivent une certaine grammaire qu'il est nécessaire de formaliser si on veut ensuite pouvoir en tirer parti.
Sauf indication contraire, les éléments obligatoires (obl) doivent apparaître une fois.

ENTRY :=	obl	entrée dans le dictionnaire (rouge)
MAIN_ENTRY	obl	entrée principale correspondant à un caractère (bleu)
COMPOUND_LIST	opt	liste de mots composés utilisant le caractère de l'entrée. (vert)


Bão 雹*. Violent orage, tempête avec pluie et grêle, ouragan.

Bão tạt ○ 悉, forte tempête. — Bão lụt ○ 潍, orage suivi d'inondation. — Đau bão 疰 ○, colique très dangereuse. — Chim bão thanh 鴆 ○ 青, autruche.

Entrée principale

MAIN_ENTRY := CHINESE_ENTRY | NÔM_ENTRY

CHINESE_ENTRY :=

QUOC	obl	Bão	une syllabe en vietnamien (quốc ngữ)
HANZI	obl	雹	un caractère chinois
et HANZI	opt		second caractère chinois précédé de « et »
*	obl	*	étoile indiquant qu'il s'agit d'un sinogramme
.	obl	.	point terminant l'entrée principale en vietnamien
FRENCH_DEF	obl		définition en français du terme

NÔM_ENTRY :=

QUOC	obl	Bão	une syllabe en vietnamien (quốc ngữ)
NÔM	obl	雹	un caractère chinois ou nôm
et NÔM	opt		second caractère chinois ou nom précédé de « et »
.	obl	.	point terminant l'entrée principale en vietnamien
FRENCH_DEF	obl		définition en français du terme

La différence entre ces deux entrées tient au fait que dans un cas il s'agit d'un mot s'écrivant à l'aide d'un caractère chinois, auquel cas une étoile est présente et dans l'autre il s'agit d'un caractère nôm.

Ceci étant, après vérification de quelques entrées, l'utilisation de cette étoile par l'auteur est contestable dans bien des cas : Bonnet (1899 :158) n'indique ni 隊, 對, 待 et 代 comme étant des caractères chinois. Pourtant ces 4 caractères existent bel et bien en chinois et en sino-japonais. Si les deux premiers n'y ont pas le même sens, ce qui pourrait expliquer qu'ils ne soient pas indiqués comme étant des caractères chinois (et donc qu'ils relèvent de l'usage du nôm), les deux suivants ont bel et bien les sens respectifs d'attendre et de siècle, génération qui sont listés dans le dictionnaire.

Définition française

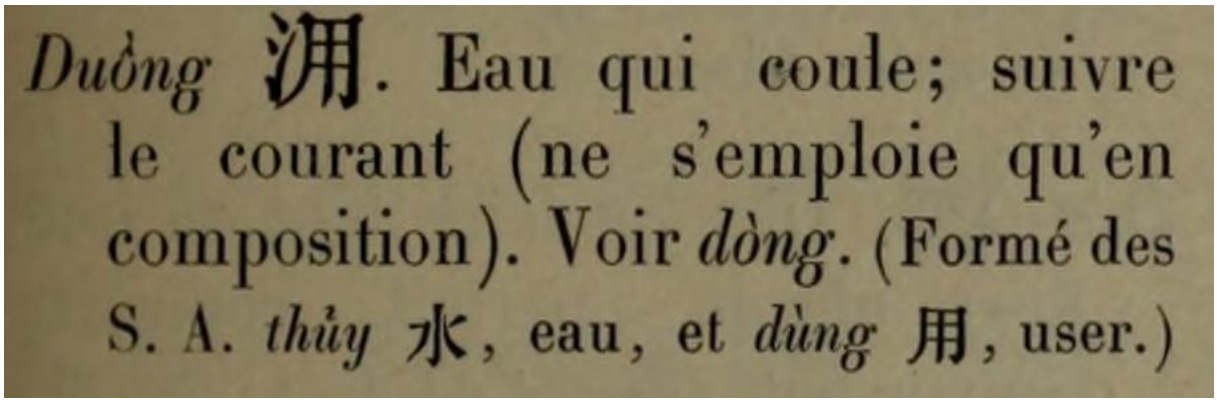
FRENCH_DEF :=
(word | word , FRENCH_DEF | word ; FRENCH_DEF)
dot
(SA_PRECISION | RENVOI SA_PRECISION)

Cette règle définie la définition de l'entrée principale en français. En plus de l'ensemble des termes qui sont listés, il y a des portions de texte ayant une signification et un emplacement spécial qu'il est intéressant d'isoler grâce des règles propres (RENVOI et SA_PRECISION) car elles constituent des informations exploitables pour l'amélioration de l'OCR ou pour la génération du fichier final.

RENVOI := Voir QUOC DOT

Le contenu du renvoi a intérêt à être isolé de façon à pouvoir le spécifier dans le fichier de sortie sous forme de métadonnées.

Précisions concernant le sino-annamite


Duồng 㵎. Eau qui coule; suivre le courant (ne s'emploie qu'en composition). Voir *dòng*. (Formé des S. A. *thủy* 水, eau, et *dùng* 用, user.)

Détails sur la formation du caractère : seconde parenthèse

Le non terminal SA_PRECISION apporte une précision étymologique ou phonétique ou concernant la composition d'un caractère. Si l'explication sur la lecture et l'étymologie, qui commencent respectivement par « En S. A., » et « Du S.A., », ne sont pas utilisables pour améliorer l'OCR, une précision qui concerne la composition d'un caractère nôm est une donnée très importante qui pourra être exploitée.

Malheureusement on note parfois des erreurs dans ces indications également (Bonnet 1899 :20) indique à l'entrée Bấng la précision suivante : (Formé des S. A. *hỏa* 水, feu, et bằng 平, paix, tranquillité.) où la clef du feu (火) qui est effectivement la clef présente dans le caractère et dont la prononciation et le sens sont bien indiqués, mais où le caractère chinois qui représente cette clef dans le texte est écrit avec celui de l'eau (水).

SA_PRECISION := open_parent SA_CONTENT close_parent

SA_CONTENT := SA_ETYMO | SA_PHON | SA_STRUCT

SA_STRUCT := (Formé des S. A. QUOC HANZI, word, et QUOC HNAZI, word) .

Liste des composés

COMPOUND_LIST :=
COMPOUND_ENTRY obl

COMPOUND_ENTRY :=

QUOC	obl(1…n)	une ou plusieurs syllabes en vietnamien (quốc ngữ)
NÔM	obl(1…n)	un nôm pour chaque syllabe du champ précédent
,	obl	virgule
FR	obl	

Pour simplifier le propos, nous n'allons pas définir ici des règles de grammaire concernant le français. Pour reconnaître ces blocs, une expression régulière comprenant l'alphabet utilisé en français et les symboles de ponctuation effectivement présents dans le dictionnaire peuvent être utilisés.

Workflow de traitement proposé

Introduction

Les tests réalisés précédemment relèvent que :

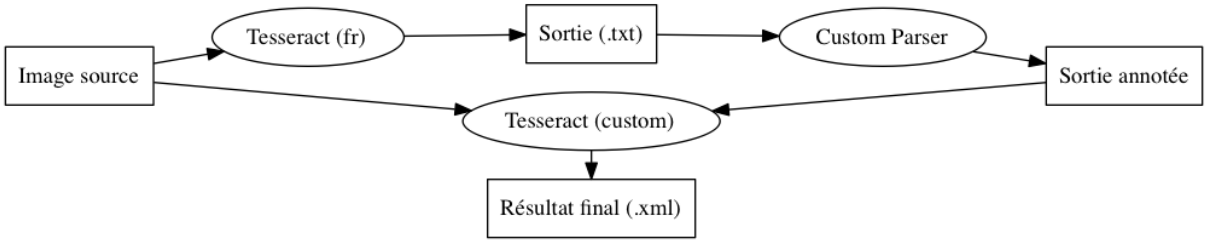
- L'analyse de l'arrangement des pages est excellente
- La reconnaissance du français est bonne
- La reconnaissance du vietnamien est mauvaise
- La reconnaissance multilingue ne donne pas de résultats satisfaisants
- En particulier le chinois doit être isolé pour fournir des résultats exploitables

C'est en prenant en compte ces points qu'une suite de traitement à l'aide d'outils existants est envisagée dans les pages suivantes. Elle constitue la cible à atteindre, mais compte tenu de sa complexité, elle n'a pas implémentée complètement.

Vue d'ensemble

Le traitement du dictionnaire est réalisé en deux passes par deux programmes, tous deux basés sur Tesseract. Un programme supplémentaire, non basé sur Tessreact, est utilisé pour la première phase de traitement.

Sur l'image suivante, les fichiers utilisés sont présentés dans des rectangles, tandis que les programmes utilisés figurent dans des ellipses.



Chaîne de traitement envisagée pour un OCR de qualité

L'image à numériser est tout d'abord traitée par Tesseract (non modifié) pour produire une sortie en utilisant le modèle de langue français. Cette sortie est traitée par un parseur (Custom Parser sur l'image) qui a pour fonction de transformer la sortie en une suite de blocs continus agencés selon la grammaire formelle du dictionnaire énoncée dans la partie précédente.

L'image source est ensuite retraitée par un programme (Tesseract Custom) basé sur l'API de Tesseract qui se sert du fichier de sortie annoté pour scanner les blocs non-français avec le modèle de langue idoine et qui applique des techniques de correction d'erreurs.

Analyse de la sortie française (Custom Parser)

Le fichier de sortie est ensuite traité par un analyseur lexical (projet **OcrOutputParser**) qui le transforme en une suite de tokens de bas niveaux. Il existe 13 tokens différents. Ces classes de premier niveau ont ensuite vocation à être remplacées par des classes de plus haut niveau indiquant la langue ou l'écriture (français, quốc ngữ ou chữ nôm) .

Nom du token	Description	Nom raccourci
CAP_WORD	Mot commençant par une lettre capitale	C
UNCAP_WORD	Mot commençant par une lettre minuscule	U
FAKE_WORD	« Mot » comprenant un symbole inexistant	假
ISOLATED_O	Un o majuscule ou minuscule ou un zéro isolé	O
STAR	Une étoile	星
NL	Saut de ligne	行
TIRET	Un tiret cadratin ou un trait d'union	T
DOT	Un point	點
COMMA	Une virgule	V
PUNCT	Un autre signe de ponctuation	P
OPEN_PARENT	Parenthèse ouvrante	開
CLOSE_PARENT	Parenthèse fermante	閉
EOF	Fin du fichier	終
pas de token	Les chiffres	aucun

Classes particulières

La casse de la première lettre d'un mot est une donnée importante : les majuscules n'apparaissent pour le français, hors nom propre, que dans les entrées principales et pas dans la liste des mots composés. Pour le vietnamien, le premier terme de chaque mot, composé ou non, dans l'entrée principale comme dans la liste des composés, est frappé d'une majuscule.

La sortie de l'OCR se retrouve parasitée par des caractères chinois mal reconnus qui peuvent être rangé dans cette classe, par exemple : ÏÊÎK.

Les mots qui commencent par une minuscule se retrouvent majoritairement dans la liste des composés, qui est souvent plus longue, quand elle existe, que l'entrée associé, ou que la plupart des entrées.

Les tokens FAKE_WORD notent des mots qui n'ont *a priori* aucune chance d'être des mots français, car ils contiennent des aberrations comme certains symboles de ponctuation ou des ligatures rares ; par exemple [lfE. Ces aberrations proviennent de tentatives de reconnaissance de mots vietnamiens ou de caractères chinois.

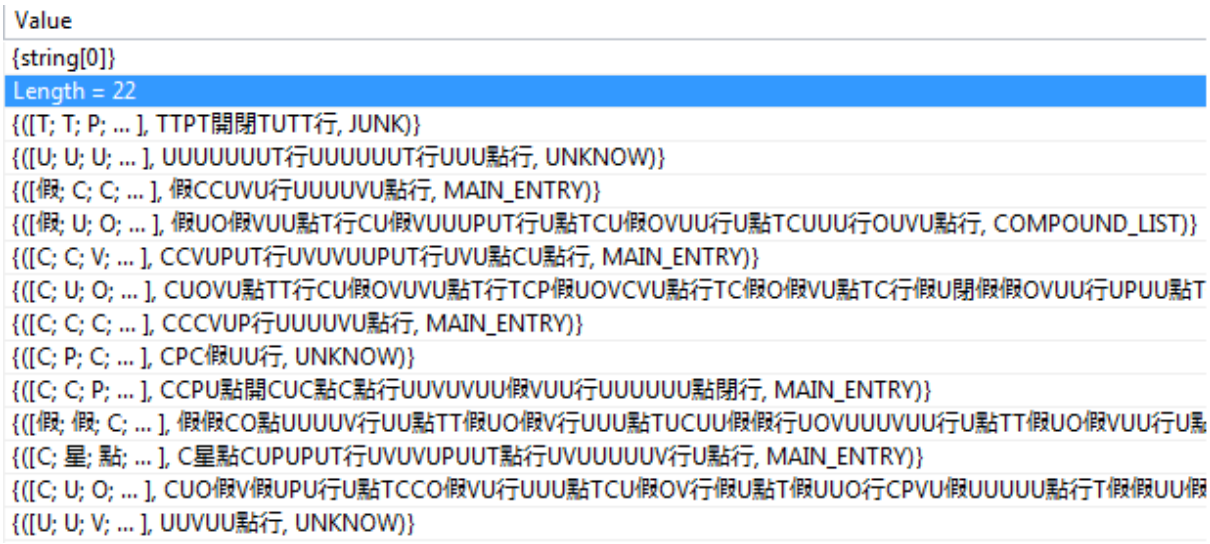
Les parenthèses ouvrantes et fermantes sont dotées de leur propre token car elles sont utilisées pour repérer et isoler la partie finale de certaines entrées principales qui peuvent se révéler intéressante. De même, les points et les virgules sont dotés de leurs tokens respectifs.

Noms raccourcis

Les noms raccourcis ont pour objectif de pouvoir représenter une suite de tokens à l'aide d'une chaine de caractères. Cela a deux applications : la première est la facilité apportée lors du développement et du débogage du programme qui crée et manipule ces suites.

La seconde est qu'il est possible de repérer des motifs dans une suite de tokens en utilisant des expressions régulières. Il est beaucoup plus facile de vérifier si une suite de tokens se conforme à un

motif donné en testant sa représentation sous forme de chaîne vis-à-vis d'un motif spécifié à l'aide d'un expression régulière elle-même sous forme de chaîne que de programmer ce mécanisme de reconnaissance en manipulant directement cette liste de tokens.



Visualisation dans le débugger de l'analyse lexicale d'une sortie d'OCR française (cf. Annexe A)

Certains noms raccourcis sont des caractères chinois. Il était convenu d'avance que chaque jeton serait représenté par un unique caractère, ce qui pose évidemment d'emblée une limite aux nombres de classes de tokens si on se limite aux caractères ASCII.

Le nommage le plus facilement compréhensible consiste à prendre la première lettre du nombre de la classe ou de des objets auxquels elle se rapporte mais il arrive vite des collisions : C pourrait servir à rendre la classe CAP_WORD, CLOSE_PARENT ou COMMA.

J'ai donc choisi d'utiliser des caractères chinois pour certaines classes, ce qui permet de limiter ces collisions tout en étant beaucoup plus explicite qu'une lettre alphabétique dans la plupart des cas. En outre, certains éléments importants, tels que les étoiles ou les pseudo-mots sont repérables bien plus rapidement de cette manière puisqu'ils se tiennent au milieu de lettres.

Catégorisation des lignes

Pour chaque ligne de la sortie de l'OCR française, après la phase de lexing, on tente d'associer à chacune de ces lignes une valeur qui correspond à la place qu'elle occupe dans le dictionnaire. Il y a quatre valeurs possibles : MAIN_ENTRY, COMPOUND_LIST, JUNK et UNKNOW.

Les deux premières (MAIN_ENTRY et COMPOUND_LIST) représentent les mêmes parties du dictionnaire que dans la partie précédente.

UNKNOW est la valeur par défaut d'une ligne, qui est utilisée jusqu'à ce qu'il lui soit assigné une des trois autres valeurs.

JUNK est affecté aux lignes qui ne sont pas à proprement parler du contenu, tel que la ligne qui indique le numéro de la page courante.

Le programme tente de rassembler les lignes entre elles quand elles appartiennent à un même bloc. Les lignes COMPOUND_LIST comprises entre deux lignes MAIN_ENTRY peuvent être fusionnées car on sait qu'elles appartiennent au même bloc. En revanche, on ne peut pas aussi facilement décider de fusionner deux lignes MAIN_ENTRY successives, car la liste des composés est optionnelle pour une entrée, il pourrait donc s'agir de deux entrées différentes.

Trois heuristiques supplémentaires découvertes à la main ont été ajoutées au programme. Cependant ce n'est pas une approche viable pour traiter l'ensemble du corpus : en effet trouver des motifs pertinents prend du temps et il faut non seulement qu'ils permettent de catégoriser au moins plusieurs lignes mais qu'en plus ils ne produisent pas de faux négatif.

L'idée envisagée, mais pas implémentée, est de passer pour cela par de l'apprentissage automatique (*machine learning*). En effet la classification est un problème que tente d'adresser l'apprentissage automatique, qui pourra détecter des motifs bien plus complexes qu'un être humain parmi ces chaînes de caractères. En outre la création d'un corpus d'apprentissage de taille significative est aisée : il n'y a pour la plupart des cas qu'à choisir entre deux la majorité du travail est déjà fait par le programme, il n'y a pas d'ambiguïté sur la classe à affecter à chaque ligne et il n'y a pas besoin d'une connaissance du domaine pour la réaliser.

Seconde phase d'OCR

La seconde phase d'OCR, représentée sur l'image « Chaîne de traitement envisagée pour un OCR de qualité » par un programme nommé *Tesseract (custom)* est réalisée par un programme basé l'API de Tesseract et plusieurs modèles de langues.

Il se base lourdement sur la sortie annotée du parseur évoquée précédemment. Le format de cette sortie reste à définir précisément, mais il s'agit d'un format « riche » qui associe la sortie brute de l'OCR à une interprétation de plus haute niveau (indication pour savoir s'il s'agit du contenu d'une entrée ou de la liste des composés ainsi qu'à une granulité plus fine, langue attendue).

Un programme qui se base sur Tesseract peut charger plusieurs modèles de langue et utiliser celui qui convient à la reconnaissance d'une ligne.

Le déroulement de la boucle principale du programme (appliqué à chaque ligne à reconnaître) est le suivant :
- OCR en français de la ligne
- avec le résultat précédent on consule le fichier annoté pour obtenir plus d'informations (1)
- on refait une analyse de ligne avec les bons paramètres (2)
- on écrit le résultat en sortie

Le fichier annoté, une fois chargé par le programme, (1) permet d'accéder à la manière d'un index aux annotations à partir du résultat de l'OCR en français. Normalement la reconnaissance, puisqu'elle est faite à partir de la même pile logicielle et avec le même modèle de langue, renvoie le même résultat, ce qui permet d'obtenir les résultats obtenus précédemment par l'analyseur de la sortie française.

C'est cependant au point (2) que se trouve toute la difficulté de création d'un tel programme, mais aussi toute la logique qui est censée permettre de fournir un résultat correct. À partir des

informations récupérées dans la sortie annotée (type de ligne, mais aussi sub-division de cette ligne en différent blocs associés à une langue), le programme réalise une seconde phase d'OCR de chacun de ces blocs.

Cette fois-ci, il utilise le modèle de langue associé à chaque bloc tout en contrôlant si la valeur renvoyée par l'API est vraisemblable. C'est à ce niveau que sont utilisées différentes techniques de détection d'erreurs.

Techniques prospectives pour la détection d'erreurs

Détection de formes vietnamiennes (quốc ngữ) impossibles par ordre alphabétique

Endroit d'application

Entrée principale en quốc ngữ.

Avant-propos

Un dictionnaire de langue occidentale à la particularité intéressante d'organiser ses entrées par ordre alphabétique. L'ordre alphabétique est variable suivant la langue donnée. Celui de l'alphabet vietnamien est sensiblement égal à celui de l'ordre utilisé en français à l'exception notable de la place de la lettre y.

L'ordre de classement s'applique aussi aux lettres diacritées. On pourrait établir un ordre total entre toutes ces lettres, mais cela prendrait beaucoup de place à écrire. Il est plus commode d'exprimer l'ordre des tons et celui des accents séparément. Les accents sont pris en compte avant les tons pour l'ordre global.

Lettre	Ordre
a	a, ă, â
e	e, ê
o	o, ô, ơ
u	u, ư

Indépendamment des lettres qui les portent, on a l'ordre suivant pour les accents : pas d'accent, brêve (˘), accent circonflexe (^) et crochet.

Les tons suivent également un ordre déterminé : pas diacritique (ngang), point souscrit (nặng), accent aigue (sắc), accent grave (huyền), tilde (ngã) et crochet en chef (hỏi). Les pages 9 à 11 de Bonnet (1899) pour les mots en Ba permettent de discerner cet ordre sans ambiguïté.

Principe de fonctionnement

L'ordre alphabétique ne peut pas être utilisé directement pour corriger directement une erreur, c'est-à-dire retrouver la forme correcte d'un mot à partir d'une forme erronée, mais il peut l'être pour détecter une erreur.

L'idée est la suivante : si un terme candidat (sortie de l'OCR) précède dans l'ordre lexicographique la dernière entrée lu et considérée comme fiable, il n'y a aucune chance pour que ce terme soit correctement reconnu puisqu'il est censé lui être postérieur dans l'ordre alphabétique. La bonne application dépend cependant de la reconnaissance de la dernière entrée. Comme celle-ci peut également être incorrecte on peut tenter de moduler en se basant sur la liste des N précédents termes, auquel cas le terme candidat sera considéré comme impossible s'il précède ces N termes selon l'ordre lexicographique.

De fait, on peut s'en servir pour élaguer la liste des candidats proposé par l'OCR pour retirer les termes qui n'ont aucune chance d'être les bons. La technique est applicable à différents niveaux : on

peut tout aussi bien détecter une proposition impossible concernant la première lettre d'un mot, que les lettres suivantes, et de manière plus intéressante les lettres diacritées représentant les accents et les tons, qui sont plus sujets à une mauvaise reconnaissance.

Exemple théorique

Reconnaître la forme đot alors que le dernier terme accepté est dôt est impossible car la lettre đ précède la lettre d dans l'ordre alphabétique.

Correction de boxing pour les termes composés écrits en chinois ou en nôm

Endroit d'application

Liste des composés : leur écriture en caractère nôm.

Avant-propos

La sortie OCR d'une suite de caractères nôm peut être constituée d'un nombre incorrect de caractères : typiquement il y a plus de caractères chinois en sortie que dans le texte d'origine. Cela s'explique notamment par le fait que les caractères nôm ne sont pas reconnus par l'OCR en langue chinoise et vont généralement donner en sortie plusieurs caractères chinois, correspondant plus ou moins aux composantes du caractère nôm.

Principe de fonctionnement

La liste des composés en vietnamien est structurée : sont donnés en premier l'écriture en quốc ngữ, plus celle en nôm et enfin une ou plusieurs définitions en français. Puisqu'on traite l'écriture alphabétique avant l'écriture logographique et que cette première écriture sépare chaque syllabe par une espace et qu'à une syllabe correspondant un caractère logographique on obtient une donnée importante : le nombre de caractères logographiques de l'expression à reconnaître.

Il faut pouvoir faire le distinguo entre ce qui ressemble à du vietnamien (comportant possiblement des erreurs) et une sortie de texte complètement fausse correspondant à la tentative de reconnaissance de caractères chinois ou nôm. Pour cela, on peut s'appuyer sur une technique de reconnaissance de langue faisant usage de trigrammes par exemple.

Avec cette information, si on connait la largeur typique X en pixels d'un vietogramme[24], il devient possible de contraindre le logiciel d'OCR à reconnaître N caractères de largeur X pour la séquence de logogrammes à reconnaître. De cette façon, le nombre de caractères en sortie, indépendamment de la qualité de la reconnaissance de ceux-ci, sera correct.

Exemple théorique

```
Dqg bào [lfE o, enseigner, conseiller. _
```

À supposer que les deux premières syllabes (Dạy bảo, ici Dqg bào) soient reconnues, la suite d'éléments séparés par des espaces qui précède la traduction française comporte le bon nombre d'éléments : 2. À savoir « [lfE » et « o ». On peut en déduire que le boxing a de forte chance d'avoir été correct pour cette partie.

[24] Néologisme forgé pour l'occasion sur la base de « sinogramme ». Une occurrence du terme apparaît également sur un forum qui ne m'est pas inconnu.

En revanche l'exemple suivant comporte un problème.

```
Khuyëzi bào  o, exhorter. —-
```

Khuyên bảo (reconnu comme Khuyëzi bào) est suivi d'un unique « o », ce qui indique la présence d'un unique caractère chinois. Or on est en présence de deux syllabes reconnues pour le vietnamien, on devrait donc avoir deux caractères nôm : on est manifestement en présence d'une erreur de boxing.

Recherche de caractères à partir de leurs composantes

Endroit d'application

Entrée principale : caractère nôm.

Avant-propos

La nature structurée des caractères chinois, en particulier ceux des composés phono-sémantiques (形聲 xíngshēng) n'a pas échappé aux chercheurs qui s'y sont intéressés et il existe plusieurs projets informatiques visant à la décrire, de façon à pouvoir s'en servir informatiquement[25]. Le fichier IDS.TXT mis en ligne par le projet Kanji Database Project décrit non seulement des sinogrammes, mais également de très nombreux caractères présents dans l'extension B d'Unicode (CJK Unified Ideographs Extension B), qui est l'endroit où sont standardisés de nombreux caractères nôm.

Ces descriptions sont faites sous la forme de séquences IDS (Ideographic Description Sequence), qui se composent d'opérateurs d'arité binaire ou ternaire situé dans la plage des points du code Unicde de U+2FF0 à U+2FFF et de sinogrammes ou de points de code

Puisqu'une information de structure peut être présente sous forme de description succincte en français pour les caractères nôm, et qu'à partir de deux composantes il existe généralement seulement un composé, on peut dans la majorité des cas retrouver le caractère décomposé.

Il faut pour y parvenir reconnaître les caractères chinois qui la composent.

Principe de fonctionnement

Pour tirer profit de cette méthode, il faut tout d'abord détecter une éventuelle parenthèse de contenu SA_PRECISION[26], puis il faut en extraire le contenu qui nous importe : les deux caractères chinois. Si au moins un des composés est un caractère nôm (ce qui peut théoriquement arriver mais qui est moins fréquent), il faut pouvoir le reconnaître ou du moins reconnaître qu'il s'agit d'un caractère nôm et éventuellement arrêter la procédure s'il n'est pas possible de continuer.

Ces deux sinogrammes en main, on peut s'en servir comme entrée pour l'algorithme décrit par Lecailliez[27] (2015) dans un travail universitaire intitulé *Méthode automatisée pour la recherche de caractères chinois complexes dans un dictionnaire électronique basée sur leur décomposition structurelle*.

[25] Lecailliez (2015 : 31-32).
[26] *Cf* « Contenu du dictionnaire ».
[27] Préférez consulter la version d'octobre 2015 plutôt que la précédente, moins complète.

Exemple théorique

Soit la function *search* qui implémente l'algorithme de recherche cité précédemment à partir des données du Kanji Database Project, on peut avoir les exemples suivants :

search(巴, 三) → 匜

search(四,本) → (vide)

Le second résultat de recherche est vide : en effet, si le dictionnaire liste les composantes de 罣 (bốn) sous la formule « (Formé des S.A., tư 四, quatre, et bổn 本, principe, racine) », ce qui est correct, les données présentes dans ISD.txt décrivent ce même caractère de la façon suivante : U+2629A 罣 ⿱罒本.

Le composant *quatre* (四) est en effet déformé à cause de sa place dans le caractère et le fichier de description le décrit à l'aide d'un caractère qui représente cette déformation. Le traitement varie selon les radicaux : 焼 est décrit ⿰火尭, c'est-à-dire sans transformation de sa clef, tandis que la clef (水) de 淡 est déformée : ⿰氵炎.

Il faut donc regarder attentivement en amont dans quels cas ces transformations sont prises en compte ou non pour savoir comment traiter les radicaux qui pourrait poser problème. On peut aussi établir une liste d'équivalence par avance et l'utiliser pour ré-effectuer une recherche si la première a été infructueuse. De manière plus générale, il faut vérifier si les décompositions fournies par les fichiers de données que l'on obtient de sources tierces sont compatibles avec l'utilisation que l'on projette d'en faire[28].

[28] Lecailliez (2015 : 36).

Sélection d'un caractère lors de la phase d'OCR

Durant le processus d'OCR, le logiciel[29] propose une liste de caractères candidats, classés par ordre de probabilité, déterminé par le modèle de langue utilisé. Dans le cas où le programme fonctionne de manière non modifiée, c'est le premier résultat de cette liste qui est produit en sortie.

Il y a deux moyens d'agir sur cette phase de sélection : en amont et en aval. L'interaction en amont consiste à utiliser une fonction de l'API qui permet de passer une fonction de probabilité qui à un caractère associe une probabilité. La manière la plus élémentaire de tirer profit de cela est de passer une fonction qui renvoie une probabilité de 100% pour le mot dont on aura déterminé qu'il est le plus probable, et 0% pour les autres.

```
/* Sets Dict::probability_in_context_ function to point to the given
 * function
 */
void SetProbabilityInContextFunc(ProbabilityInContextFunc f);
```

L'interaction en aval consiste à consulter la liste des caractères candidats et de choisir nous-même celui qui semble être le plus probable. Elle a l'inconvénient notoire de ne pas fonctionner si le mot cible n'est pas inclus dans cette liste (auquel cas le résultat renvoyé sera incorrect, et ne fournit donc pas d'amélioration par rapport à une non intervention).

Algorithme

C'est à ce niveau que je propose d'appliquer le traitement suivant :
- la syllabe écrite en quốc_ngữ est transformée en une approximation phonétique (syllabe de contrôle)
- les N premiers termes candidats proposés par le logiciel d'OCR sont recherchés dans un dictionnaire de chinois ancien
- une approximation phonétique est générée pour chacun de ces termes
- ces dernières approximations sont comparées à la syllabe de contrôle
- un choix est fait
(- il est possible de répéter le processus pour un terme ou une expression de plusieurs syllabes)

L'algorithme d'approximation est à définir à la fois pour le vietnamien, et pour le chinois moyen. La méthode comparaison également, et c'est l'expérience qui déterminera en partie quel est le plus pertinent à utiliser. Enfin la méthode de sélection, présentée comme le « choix » final est également à définir : il pourrait s'agir de sélectionner le premier terme candidat dont la comparaison avec la représentation de la syllabe de contrôle dépasse un certain seuil, ou de sélectionner le caractère ayant le meilleur score de comparaison parmi les N premiers, par exemple. Dans tous ces cas, différentes méthodes vont être esquissées, et testées, pour voir ce qui en pratique fonctionne le mieux.

Un défaut évident est inhérent à l'utilisation d'un dictionnaire : il peut manquer des éléments pour pouvoir générer une approximation du terme de contrôle. Pour le contourner, on peut se tourner vers d'autres sources[30].

[29] Plus précisément son API expose les termes candidats.
[30] Je détaillerai plus loin dans ce document.

De manière moins prégnante, un caractère chinois peut être associé à plusieurs lectures, que ce soit en chinois moyen ou en vietnamien. Ce défaut est mineur car le cas se présente peu souvent, mais il est à garder en mémoire.

Enfin, il existe un problème qui ne pourra pas être résolu : la méthode ne fonctionne que pour les caractères chinois puisque d'une part les caractères annamites (nôm) n'existent pas en chinois[31], ce qui empêche de créer un dictionnaire faisant l'association entre un caractère et sa lecture et donc d'en tirer une représentation phonétique approximée, et d'autre part, les mots représentés sont du vietnamien et n'ont de fait pas de racine chinoise.

Il est toutefois possible d'utiliser un dictionnaire faisant l'association nôm -> lecture sino-annamite. Un tel dictionnaire existe, avec un nombre restreint de caractères : les données sont extraites du site chunom.org. C'est d'ailleurs sur ces données que seront testés les algorithmes d'approximation et de comparaison entre prononciation annamite et chinoise pour un même caractère.

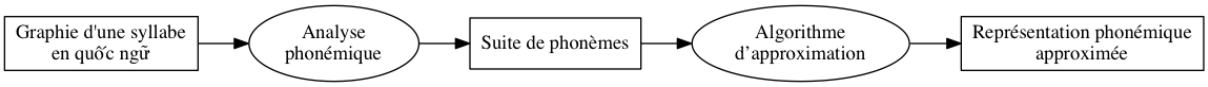
Penchons-nous à présent sur la formalisation de cette méthode et son cadre théorique.

Il faut d'un côté un moyen de transformer une syllabe vietnamienne en une approximation phonétique, et de l'autre générer une approximation relativement comparable à partir d'une source sur le chinois moyen. Parce qu'il y a un niveau d'indirection supplémentaire dans le second cas, il faut traiter ces deux processus séparément.

Génération d'une approximation phonétique pour le vietnamien

On prend pour hypothèse de travail que les variations dialectales sont hors de portée de ce document. En outre, la réalisation effective phonétique des phonèmes est considérée comme n'ayant pas d'importance dans le cadre de notre approximation.

L'écriture en quốc ngữ n'est pas bijective avec les phonèmes qu'elle sert à représenter : il existe 68 graphèmes pour 50 phonèmes[32]. Il faut donc réaliser une analyse phonémique pour transformer une chaîne de caractères en une représentation (elle aussi sous forme de chaîne de caractère) phonémique. C'est cette représentation qui sera ensuite transformée en une approximation phonétique par l'algorithme idoine.



C'est la suite de traitement théorique qui sert à générer une approximation à partie d'une graphie. Dans la pratique, comme certains phonèmes sont représentés de manière consistante par le même graphème et que certains phonèmes seront unifiés dans une même classe dans la seconde partie du traitement, j'ai choisi de ne pas réaliser une analyse phonémique exacte dans la première phase de traitement : le développement est accéléré du fait qu'il n'y a pas à prendre en compte certaines différence ni à devoir gérer des caractères de l'alphabet phonétique international qui n'existent pas sur les claviers standards.

[31] Hormis quelques télescopages, tel que 喃.
[32] Huynh (2013: 13).

Le processus d'analyse reste le même, mais le résultat intermédiaire n'est pas une suite de phonèmes représentées par des symboles de l'alphabet phonétique international mais par une structure de données faisant usage de lettres latines.

Génération d'une approximation phonétique pour le chinois moyen

Le problème du chinois moyen est que contrairement au vietnamien, il n'en existe pas de traces dans une écriture phonétique ou phonologique. Il faut donc travailler à partir de reconstructions, basées sur des sources indirectes. En outre, le chinois moyen n'est ni la même langue que le vietnamien, ni daté de la même époque. Il faut donc prendre en compte ce fait lorsqu'on tente de comparer des éléments entre ces deux langues.

Ces sources indirectes sont les langues sinoxéniques[33] qui ont fossilisé un certain nombre de termes dans leur langue : le sino-coréen, le sino-japonais (lectures KAN-ON et GO-ON) et le sino-annamite. Ainsi que les langues chinois elles-mêmes (langue commun, cantonnais, etc.) Ces données ont permis à des chercheurs comme Karlgren, Maspéro et plus récemment Baxter et Sagard de proposer des reconstructions du chinois moyen (*middle chinese*) et du chinois ancien (*old chinese*).

Concrètement, utiliser ces reconstructions suppose de prendre en compte que le vietnamien a évolué, selon des règles phonétiques qui lui sont propre, entre le moment où les termes chinois ont été empruntés et la rédaction du dictionnaire étudié. Dans l'idéal, toutes les lois phonétiques du vietnamien de l'époque d'emprunt vers le vietnamien actuel sont appliquées aux reconstructions afin de pouvoir travailler sur une base de termes comparables. Il faudrait en outre prendre en compte le passage du chinois au vietnamien de l'époque, qui entraîne lui aussi des modifications propres.

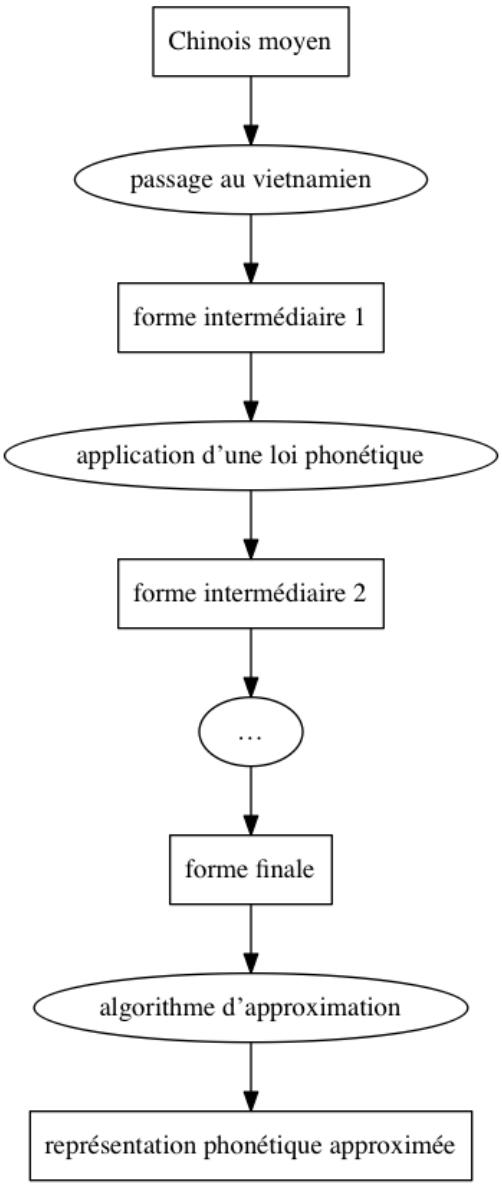
Le texte de Maspero comprend de nombreuses illustrations de ce phénomène[34] :

> « *Le sino-annamite a également* ó *; cet* ó *est plus difficile à expliquer. On a déjà vu ann.* ó *rendant le chinois* uá *dans les mots à finale* uái *; je crois que dans les deux cas il s'agit d'un fait de même genre: les Annamites, ne trouvant pas dans leur langue l'équivalent des diphtongues chinoises, ont rendu de leur mieux le timbre grave de l'ensemble qu'ils étaient incapables de décomposer en ses éléments*[35]. »

En plus de la complexité des transitions qui concernent les voyelles et les consonnes, les langues sources et cibles sont tonales, il se rajoute donc également des problématiques complexes d'interaction tonale.

Si on voulait comparer au plus près les syllabes vietnamiennes actuelles avec les syllabes vietnamiennes qu'on obtiendrait après application de différentes lois phonétiques, on se retrouverait avec une chaîne de traitement ayant la forme :

[33] Karlgren (1922 : 4).
[34] Maspero (1912 : 102).
[35] *Karlgren, Phonologie chinoise, p. 617.*



Bien entendu trouver ou établir une liste exhaustive de lois phonétiques, à supposer qu'elles existent dans la littérature, nécessiterait plus de recherches documentaires que ne le permet la rédaction de ce mémoire, en plus d'une bonne connaissance de la langue vietnamienne pour lire les articles publiés dans celle-ci.

On en arrive donc à la simplification suivante : on compare directement les reconstructions du chinois moyen aux syllabes vietnamiennes en supposant que dans le cas général les évolutions décrites précédemment affectent peu la différence entre syllabes dans la langue cible (vietnamien) si ces différentes étaient notables dans la langue source (chinois moyen).

Approximation phonétique

Pour pouvoir comparer les syllabes entre deux langues différentes, il faut une base de travail commune. Le principe général utilisé pour la création de cette base de travail est le suivant : des classes de phonèmes (représentées par des lettres capitales) proches ont été créées à partir des

graphèmes[36] de la langue vietnamienne. D'autres classes ont été utilisées pour le chinois moyen, certaines étant les mêmes que pour le vietnamien. Puisque ces ensembles ne pas exactement les même, on peut déjà prédire qu'il n'y aura pas de correspondance totale entre les approximations dans ces deux langues.

Les classes utilisées sont listées ci-dessous. Elles sont énoncées à raison d'une par ligne sous la forme /CLASSE/ graphème … graphème.

Consonnes et semi-voyelles

/B/ B
/K/ C K KH QU
/G/ G
/D/ Đ D
/H/ H (aspiré)
/L/ L R
/M/ M
/N/ N
/F/ PH
/X/ S X (sh)
/T/ T TH
/V/ V
/Ñ/ NG NGH
/Ø/

Consonnes et semi-voyelles spécifiques au chinois

/B/ P
/Č/ TSH
/Y/ I J Y
/W/ U W
/N/ NY
/D/ DZ

Voyelles

/a/ a
/é/ e
/i/ i y
/o/ o ô
/e/ ơ
/u/ u
/ü/ ư

[36] Normalement ce sont les phonèmes de la langue qui auraient dû être utilisés, mais comme le programme manipule en entrée des graphèmes, il était plus simple de procéder comme cela.

Méta-classes pour les finales

/S/ (sourdes : p t k)
/N/ (nasales : m n ng)
/Ø/

La classe /Ø/ représente l'absence d'un phonème.

Note d'implémentation

Le code utilise la même table de conversion pour convertir les graphèmes d'une langue en classe approximée de phonèmes, c'est qui est la cause d'au moins un bug.

Code de comparaison (Vietdex)

Le code approximant une syllabe est inspiré de l'algorithme Soundex[37]. Ce dernier ayant été créé pour l'anglais britannique et ayant pour objectif la création d'index, il ne peut pas être réutilisé directement ici.

Premier code envisagé

Le premier code envisagé avait le format suivant :

C1 : Initiale / Voyelle principale / Nombre de voyelles / Catégorie de la finale

Où *Initiale* est la classe de l'initiale de la syllabe, *Voyelle principale* celle de la voyelle principale. Le code de *Catégorie de la finale* est une méta-classe prise parmi /S/ (sourdes : p t k) /N/ (nasales : m n ng) et /Ø/. Son objectif est de rendre compte du type de finale en chinois moyen. Enfin le *Nombre de voyelles* indique combien de voyelles et semi-voyelles sont présentes dans la syllabe.

Exemples

Les exemples sont donnés, à raison d'un par ligne et accompagnés de commentaires, sous la forme : sinogramme vietnamien (reconstruction TON) -> code_vietnamien / code_chinois

空 không (kh.uwng A) -> KO1N / KU2N
Il s'agit ici d'un exemple qui montre bien l'usage de chacun des champs. La syllabe est prononcée de manière assez proche dans les deux langues et cela se traduit effectivement par la moitié du code qui est identique dans les deux cas.

姑 cô (k.u A) -> KO1Ø / KU1Ø
Cet exemple ressemble beaucoup au premier, à ceci près qu'il y a le même nombre de voyelles. On commence à apercevoir un problème avec ce système de codage : la différence du type de voyelle, moins importante qu'une différence d'initiale ou de finale est pourtant placée avant celle de la finale, ce qui peut compliquer l'algorithme de comparaison qui s'en servira.

爲 vì (hj.we A) -> VI1Ø / HE?Ø
Certaines reconstructions posent des problèmes d'interprétations. En l'absence d'une copie de *Old Chinese : A New Reconstruction* de Baxter et Sagart il m'est difficile d'être sûr de ce que sont censés indiquer précisément certaines reconstructions.

[37] C'est pourquoi dans l'algorithme d'approximation du vietnamien est nommé Vietdex dans les programmes et dans leurs sorties.

Second code envisagé

Le premier code comporte un point gênant : la comparaison des codes en utilisant l'ordre lexicographique n'est pas possible car certains champs, à mon sens plus importants, sont placés en fin de chaîne.

En outre, l'utilisation de méta-classes pour les finales semble être un peu trop imprécise. Le code envisagé est le suivant :

C2 : Initiale / Finale / Nombre de voyelles / Voyelle principale

L'indication concernant la finale a été déplacée à la seconde place afin de rendre compte de son importance. La voyelle principale et le nombre de voyelles ont été permutés. Enfin c'est la classe de la finale qui utilisée et non plus une méta-classe. De plus, les classes de voyelles sont écrites en minuscules afin de les reconnaître plus facilement.

Exemples

Le format des exemples est le même que précédemment.

雪 tuyết (s.jwet D) -> TS3e / SS2e

卬 ngang (ng.ang A) -> ÑN1a / ÑN1a

於 ở ('.u A) -> ØØ1e / ØØ1u

Code final

Les exemples de code donnés précédemment ont été réalisés à la main. La complexité de génération informatique n'a pas été prise en compte. En particulier, pour le même problème d'interprétation des reconstructions soulevé précédemment et parce que la structure syllabique du chinois moyen est plus compliquée que celle de vietnamien, il serait nécessaire d'avoir un analyseur pour ces reconstructions.

Le travail pour le réaliser, au vu des efforts mis en œuvre pour celui du vietnamien, n'est pas compatible avec le temps imparti. Aussi j'ai simplifié le code de manière suivante :

Consonne initiale / consonne finale

Test de l'automate d'analyse

La première expérience, qui consiste à comparer les approximations phonologiques entre le vietnamien et le chinois moyen a été réalisée sur un échantillon de 370 caractères. Lors de cette expérience l'automate n'était pas tout à fait finalisé, certaines analyses de syllabes ne sont donc pas correctes. En outre, toutes les initiales et finales du chinois classique n'ont pas été affectées à une classe, il manque donc aussi des informations à ce niveau-là. Dans tous les cas, ces manquements sont décomptés dans une catégorie séparée.

Les 370 caractères nôm du niveau 1 et 2 listés par le site chunom.org[38] ont servi de corpus. Parmi eux, 254 sont des caractères ayant un point de code Unicode inférieur à U+9FC3 et donc appartenant au

[38] Nguyen, Trung (2012-2015). Chunom.org

plan de base multilingue (BMP). Les 116 caractères restant sont donc très probablement[39] des caractères nôm non chinois.

Le fichier de sortie qui est discuté dans la suite est disponible ici : https://raw.githubusercontent.com/titanix/m2_vietnamese_analysis/master/VietPhonology/bin/Debug/output370.html

Classes de dénombrement

L'objectif de cette expérience est de voir si les approximations générées via le vietnamien et via le chinois moyen pour un même caractère sont les mêmes. Dans l'idéal elles doivent être identiques pour tous les éléments étudiés, mais on sait d'avance que ça ne sera pas le cas (ne serait-ce qu'à cause des différentes classes utilisées pour ces deux langues) : il s'agit donc de voir dans quelle mesure c'est le cas ou non. On va donc compter les caractères pour lesquels ces deux approximations sont identiques.

Pour faire ce décompte, on utilise quatre classes différentes :
- 1 : les deux approximations sont égales
- ~ : les deux approximations sont presque égales (cf. infra)
- ≠ : les deux approximations sont différentes
- m : une des approximations est manquante

Deux annotations supplémentaires existent : b et « vide », qui se superposent aux classes.

La classe 1 est affectée à deux approximations strictement égales, par exemple KÑ et KÑ.

La classe ~ est affectée aux couples d'approximations dont seul l'initiale varie et si ces deux initiales ont des caractéristiques linguistiques proches. La liste des couples d'initiales permettant de répondre à cette condition est fermée : T/D, K/G, F/V. Le premier groupe est formée de consonnes occlusives dentales, le second d'occlusives vélaires et le troisième de fricatives labio-dentales[40] ; dans les trois cas, c'est la distinction entre sourde et voisée qui est considérée comme peu pertinente.

La classe ≠regroupe tous les couples qui diffèrent et qui ne peuvent pas être rangés dans la classe précédente.

La classe m (comme manquant) note les caractères pour lesquels une ou l'autre des approximations n'a pas pu être générée en l'état des programmes actuels. Le plus souvent il s'agit de la reconstruction en chinois moyen mais dans quelques cas, c'est l'automate d'analyse de syllabes vietnamiennes qui produit un résultat erroné ou manquant. Cette classe n'est affectée que lorsqu'une reconstruction pour le chinois moyen existe, sinon l'entrée n'est pas annotée (annotation « vide »).

En plus de cela, les bugs de l'automate « b » sont relevés pour les cas où une des 4 classes devraient normalement être affectées.

[39] Il existe des caractères chinois utilisé en chinois hors du BMP, y compris dans les données de Baxter & Sagart.
[40] Si on compare la consonne de référence de ces classes.

Analyse des résultats

Sur les 254 caractères du BMP, 162 se sont vus affectés une classe, 92 ont donc une annotation « vide ».

Classe	Dénombrement	Pourcentage (sur 162)
1	62	38,3
~	29	17,9
≠	39	24,0
m	32	19,8
Classes cumulées	**Dénombrement**	**Pourcentage (sur 130)**
1 et ~	91	70,0
~ et ≠	68	52,3

Ce tableau montre que l'expérience est grandement améliorable d'un point de vue quantitatif : environ 1/5 des données exploitables n'ont pas pu être traitées, principalement parce que l'initiale d'un mot en chinois moyen n'a pas encore été affectée à une classe de phonèmes ou parce qu'une méthode pour définir la consonne finale est manquante.

La performance pour cette expérience dépend de la façon dont doit être traitée la classe ~. Si on considère que groupée à la classe 1 elle permet de faire un rapprochement entre la lecture vietnamienne et une reconstruction du chinois moyen, alors le résultat de l'expérience est très positif : la méthode permet dans 70% des cas de rapprocher des prononciations liées.

Si au contraire on considère que la classe ~doit être groupée avec la classe ≠, les résultats sont notoirement moins bons : un rapprochement ne peut être fait qu'à peine dans un cas sur deux, ce qui n'est guère mieux que de tirer à pile ou face.

Dans un petit nombre de cas, il existe une reconstruction en chinois moyen pour un caractère chinois donné mais la forme référencée par les données de travail n'est pas la même que celle présente dans le corpus Baxter-Sagart. Les cas concernés sont : 别別, 爲為, 学學, 払拂 et 体體 où le premier caractère est celui présent dans les données nôm et le où second est le caractère présent dans le dictionnaire de chinois moyen.

Implémentation de l'expérience

Le code du programme et les données utilisées se trouvent dans le projet F# **VietPhon**. Ce projet fait usage de la bibliothèque réalisée par le projet **VietSyllableTransducer**. L'exécutable généré prend en entrée un fichier de donnée[41] qui contient des caractères nôm ou chinois et leur lecture. Les lignes impaires contiennent un caractère tandis que les lignes paires contiennent la lecture en quốc ngữ du caractère de la ligne précédente.

Pour chaque caractère le programme analyse sa prononciation vietnamienne, en produit une approximation phonétique, cherche une reconstruction du chinois médiévale correspondante, en génère une approximation si elle existe et affiche ces données en plus des données d'origine.

[41] Actuellement le chemin relatif est *hardcodé* vers le fichier de donné livré avec le projet dans le dossier *Data*.

Programmes réalisés

Transducteur d'analyse d'une syllabe vietnamienne

Une des techniques proposée précédemment pour améliorer la reconnaissance des caractères chinois consiste à vérifier si leur prononciation en vietnamien, écrite en quốc ngữ (lettres latines), correspond approximativement à la reconstruction en chinois moyen du caractère chinois candidat de la sortie d'OCR.

Pour cela nous avons besoin de générer une approximation de la prononciation de la syllabe vietnamienne, ainsi que de la reconstruction en chinois moyen. La génération de l'approximation utilisée dans la comparaison est indépendante de la syllabe elle-même : on peut imaginer de comparer des syllabes sur plusieurs critères.

Il faut donc un moyen de faire une analyse linguistique de la syllabe qui servira d'entrée à l'algorithme d'approximation phonétique.

Relation graphème-phonème

La relation entre les graphèmes de l'écriture vietnamienne en quốc ngữ et les phonèmes de la langue n'est pas une bijection : il y a plus de graphèmes que de phonèmes à représenter[42]. En outre, il y a des cas particuliers où certaines lettres représentent des phonèmes bien différents suivant leur contexte, par exemple la lettre « o » qui peut servir à noter /ɔ/ dans le cas général ou /u/ dans certains cas particuliers[43].

Pour avoir une représentation de la prononciation du mot, il faut donc pouvoir retrouver la suite de phonèmes d'une syllabe à partir de son écriture en lettres latines (avec l'hypothèse que les réalisations phonétiques de ces phonèmes ont peu d'importance, car si ce n'est pas le cas, il faut bien entendu faire des traitements supplémentaires).

En outre, puisque l'algorithme d'approximation phonétique se base sur des éléments structurels de la syllabe (ex : consonne finale), il faut également transformer la forme graphique en une représentation structurée de la syllabe.

En programmation orientée objet cette représentation structurée prend la forme d'une classe qui représente une syllabe et permet l'accès aux éléments de sa structure de manière indépendante.

Transducteur à états finis

Comme l'indique Hippisley dans un chapitre de *Handbook of Natural Language Processing*[44], « the favored model for handling morphonology in the orthography, or morphology-based orthographic spelling variation, is a specific type of finite state machine known as a finite state transducer (FST). » Nous ne sommes pas intéressés ici par un découpage en morphèmes (une syllabe équivaut à un morphème en vietnamien) mais en phonèmes, mais c'est la même approche qui est utilisée : l'analyse par un transducteur fini.

[42] Huynh (2013 : 13-17).
[43] Huynh (2013 : 15).
[44] Hippisley, Andrew (2010 : 33).

Un tel transducteur, qui travaille sur un langage régulier, peut être implémenté par deux approches théoriquement équivalentes : la reconnaissance par une expression rationnelle (expression régulière) qui fait usage de parenthèses capturantes ou par un automate à états finis.

Si l'approche par expression rationnelle a pour avantage sa concision — elle est exprimée en une chaîne de caractères et la plupart des langages de programmation en permettent l'utilisation en quelques lignes de code — l'écriture est en revanche mal aisée. Il est difficile d'écrire et de déboguer une expression rationnelle et il ainsi probable de tomber dans un des deux problèmes que la formalisation d'un langage régulier peut rencontrer : (1) ne pas reconnaître certains mots appartenant au langage ou (2) reconnaître des mots qui n'appartiennent pas au langage qu'on tente de modéliser.

Puisque pour l'analyse structurelle d'une syllabe nous savons *a priori* qu'il s'agit d'un mot du langage, le second problème est mineur comparé au premier.

J'ai donc choisi de passer par l'implémentation d'un automate fini déterministe. Contrairement à une expression rationnelle, il peut se représenter facilement de manière graphique, ce qui est un avantage non négligeable pour la facilité de compréhension. Les automates finis déterministes et non déterministes ayant la même puissance de reconnaissance (ils reconnaissent tous deux les langages de type 3 de la hiérarchie de Chomsky), j'ai préféré travailler sur un automate déterministe, puisque celui-ci impose de réfléchir pour chaque état quel est l'état suivant pour tous les symboles de l'alphabet du langage.

Représentation graphique

La représentation graphique de l'automate sur papier devient vite difficilement lisible et comme une image est nécessaire pour illustrer ce mémoire, une version informatisée a été réalisée à l'aide du logiciel Graphviz. Ce programme prend en entrée un fichier texte qui décrit un automate à l'aide d'un langage spécifié dans sa documentation.

Le fichier image, trop large pour être placé dans ce document, est accessible à l'adresse : https://unixorn.azurewebsites.net/dl/viet_syllable_analyzer.png

Cependant, il reste à implémenter l'automate dans un langage de programmation proprement dit, afin de pouvoir s'en servir[45]. Indépendamment de la technique utilisée pour sa réalisation effective, un nouveau problème se pose : comment garantir que l'automate programmé est bien conforme à sa spécification ? (c'est-à-dire à dire celle du fichier utilisé par Graphviz pour sa description graphique). En effet, chaque modification de la spécification doit avoir son pendant dans le code de l'automate et chaque erreur humaine dans l'implémentation de l'automate vis-à-vis de la spécification peut entraîner un problème qui peut facilement passer inaperçu.

Pour le régler, j'ai choisi d'utiliser le fichier de spécification pour générer le code (méta-programmation). De cette façon, chaque changement apporté à la spécification est immédiatement

[45] J'aurai également pu passer par une solution tierce toute faite. Par exemple Unitex, qui est un ensemble de programmes et un IDE permettant notamment de créer, d'utiliser et de visualiser des automates. Le format de fichier natif de ce logiciel est cependant peu *user-friendly*, puisqu'il faut spécifier à la main la position (dans un système de coordonnées cartésiennes) de chaque état de l'automate, ce qui devient vite très embêtant à gérer. Graphviz a pour avantage de déterminer automatiquement le positionnement de chaque état.

visible graphiquement (une nouvelle image est générée), et le code principal de l'automate est facilement mis à jour pour être conforme aux modifications.

J'ai donc écrit un compilateur qui permet de lire le fichier d'entrée destiné à Graphviz (dans une version plus limitée du langage que celui-ci supporte) afin de générer les états et les transitions de l'automate. Les actions d'écriture en sortie qui font de cet automate un transducteur sont écrites à la main[46].

Automate d'analyse des syllabes

Le symbole est ajouté à la fin de la chaîne avant ø le début de son traitement. De cette façon, il est possible de distinguer les cas où le traitement est fini, et donc que des pans de la structure de la syllabe doivent être fixés, des cas où le traitement continue.

Comme les deux premières parties (consonne initiale et semi-consonne préfrontale) sont optionnelles, à la différence du noyau vocalique, qui n'est suivi que d'une partie optionnelle (la consonne finale), j'ai décidé d'analyser la chaîne en commençant par l'arrière. Lorsque la chaîne est passée à l'automate pour analyse, elle est tout d'abord renversée puis le symbole ø y est ajouté à la fin. Le traitement proprement dit est ensuite lancé.

Les tons ne sont pas gérés par l'automate : en effet l'existence de nombreuses diacritiques qui peuvent être placées sur les voyelles ajoute 64 versions différentes de ces lettres en plus des lettres de base (a, e, i, o, ơ, u, ư, y), et donc autant de transitions à gérer pour chaque états de l'automate. Les voyelles représentées avec des accents et des tons sont considérées comme étant similaire à leur version non tonale, sauf pour les cas relevés au paragraphe suivant.

Je considère que ơ et ư ne sont respectivement pas les lettres « o » et « u » diacritées car le phonème auquel ils correspondent est vraiment différent[47] des phonèmes représentés par « o » et « u ». En effet, comme l'indique Huynh[48] « u » note /u/ tandis que « ư » note /ɯ/ et « o » et « ô » notent respectivement /ɔ/ ou /o/ tandis que « ơ » note /ɤ/. Cette distinction est importante pour la phase suivante de génération d'une représentation approximative de la prononciation et doit être conservée. Pour une raison similaire, la lettre « ê » est gérée à part.

Structure de l'automate

L'automate contient plusieurs états finaux, dont le plus utilisé est nommé *End*. Le compilateur génère un état appelé *__END* qui est atteint lorsqu'aucune transition n'a été atteinte, ce qui est le symptôme d'un problème de reconnaissance de l'automate (lors de la phase de développement) ou qu'on tente d'analyser une chaîne de caractères qui n'est pas une syllabe (lorsque l'automate sera considéré comme développé et « sans bug »). Lorsque cet état est atteint, l'état qui a mené à cela est indiqué dans l'objet de syllabe renvoyé par l'analyseur (on peut alors l'afficher dans la sortie du programme), ce qui permet de débogueur l'automate.

[46] Une approche par convention de nommage des états auraient pu éviter cette étape, au prix d'une étape supplémentaire dans la génération de code et de la définition de ladite convention, dont les prémisses apparaissent d'ailleurs dans le nom de certains états (OutGH par exemple).
[47] Le critère pour unifier ou distinguer ces voyelles est mon jugement en tant que locuteur non natif.
[48] Huynh (2013 : 15).

Afin de s'assurer que le cas de chaque transition pour chaque symbole ait été réfléchi, j'impose que chaque état non final contienne une transition étiquetée par un symbole de l'alphabet. Comme toutes les transitions n'ont pas de sens pour tous les états (par exemple 'q' ne peut jamais se retrouver autre part qu'en tant que consonne initiale), il existe un « état poubelle » nommé *Impossible* vers lequel convergent les transitions qui n'ont pas de sens. Si l'état poubelle est atteint, cela est consigné dans l'objet de syllabe renvoyé par l'analyseur afin que le code appelant puisse se rendre compte que la chaîne n'est pas une syllabe valide. Le compilateur de l'automate génère alors une ligne qui génère elle-même un warning lors de la compilation de l'analyseur lorsqu'une transition n'est pas définie pour un état non final[49].

Implémentation

L'automate d'analyse d'une syllabe vietnamienne est implémenté par un programme C#. Une partie du code de celui-ci est générée par un compilateur écrit en F# qui utilise le fichier .dot (format Graphviz) qui décrit l'automate. Le fichier exécutable de l'automate peut être référencé par d'autres projets pour être utilisé en tant que bibliothèque ; c'est ce qui est fait avec le programme F# qui effectue les expériences liées à l'algorithme d'approximation phonétique.

F# est un langage multi-paradigme (fonctionnel, impératif et orienté objet) à typage statique fort dérivé d'OCaml qui s'exécute sur le Common Language Runtime (CLR). C# est un langage orienté objet à typage statique fort qui fonctionne également sur le CLR.

Compilateur de graph

Le compilateur de graph est implémenté à l'aide de FsLex et FsYacc. Un fichier .fsl lu et compilé par FsLex permet de créer un analyseur syntaxique (aussi appelé lexer ou tokenizer). Cet analyseur permet de générer une suite de token qui sera lue par le parseur, créé à l'aide d'un fichier .fsy et de FsYacc.

La grammaire reconnue par le parseur n'est pas définie *a priori*, le code source sert de définition *ad hoc*. Elle est un sous ensemble de celle reconnue par Graphviz. De manière générale, seuls les tokens et les règles de grammaire nécessaires à la génération de l'automate ont été implémenté.

En particulier, l'association de plusieurs états vers un état unique (état1 -> { état2 état3 } par exemple) n'est pas supportée puisque l'automate se voulant déterministe, en aucun cas un état ne peut mener vers plusieurs autres à l'aide de la même transition.

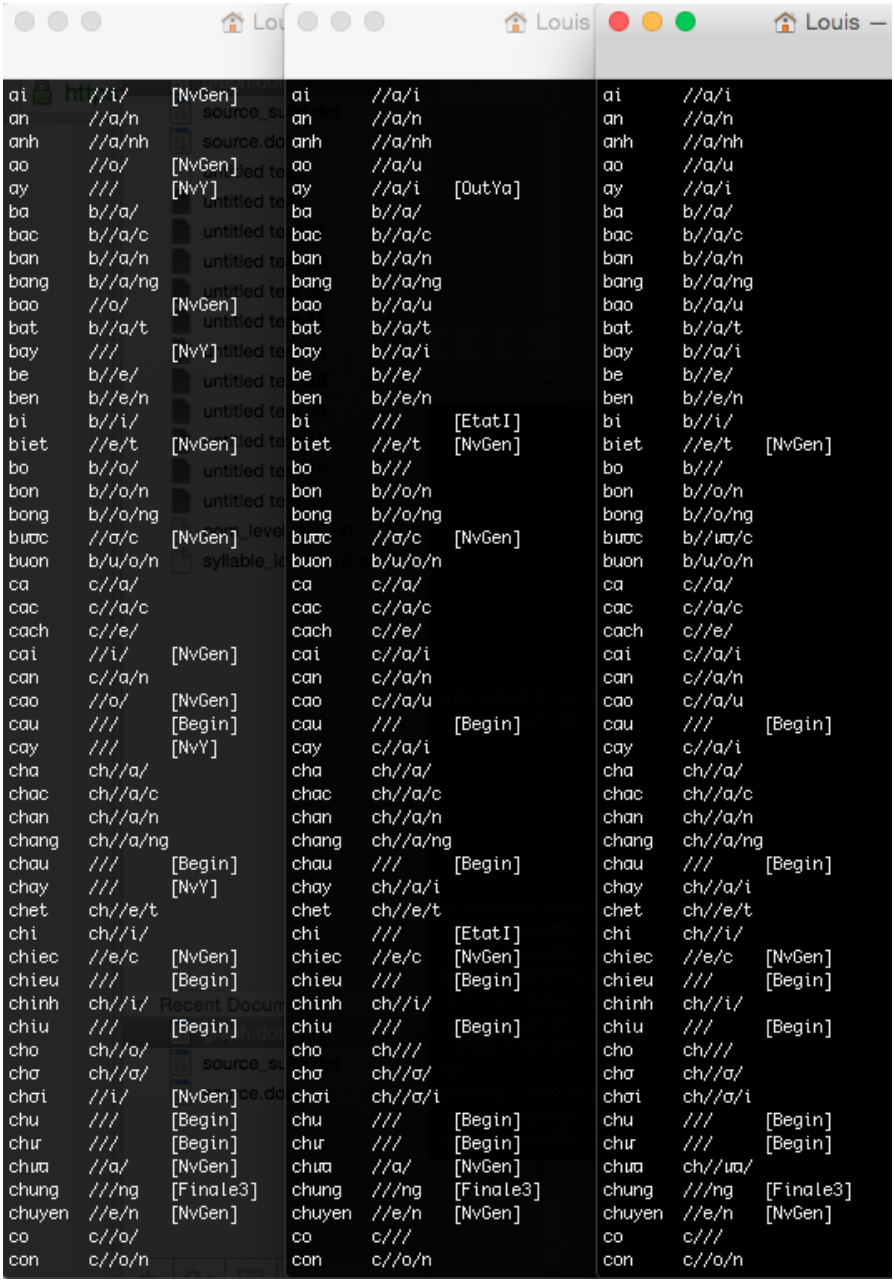
La réciproque, c'est-à-dire les règles de la forme { état1 état2 } -> état3 sont en revanche supportées et sont très utiles pour réduire le nombre de transitions à écrire. Les règles de la forme état1 -> état2 -> état3 ne sont pas supportées.

Lors de la phase de tokénisation, l'analyseur s'attend à trouver une ou plusieurs lettres de l'alphabet comme étiquette (label) d'une transition. Les lettres de l'alphabet reconnu sont transformées en jetons QUOC_LETTER, qui sont associés à un caractère. De ce fait, les états ne peuvent pas être nommés à l'aide d'une unique lettre appartenant à l'alphabet des symboles reconnus car leur

[49] L'infrastructure de génération des warnings est en place mais l'utilisation de l'état « poubelle » n'est pas implémentée.

reconnaissance en tant que token QUOC_LETTER entraînerait des erreurs dans la phase d'analyse syntaxique.

Développement itératif



On voit sur cette capture d'écran la progression de l'implémentation de l'automate. Trois terminaux sont ouverts. Le plus récent est à droite (T3), le plus ancien à gauche (T1).

Sur la première ligne sur T1, la syllabe « ai » n'est pas reconnue, et de plus sa finale « i » est considéré comme le noyau vocalique ce qui est faux. En T3 elle est correctement analysée (modulo la prochaine note de bas de page). À la cinquième ligne, « ay » est correctement analysé, sa finale étant

unifiée avec « i »[50]. Notons que le « a » correspond à un phonème différent dans ces deux cas, ce qui se reflète dans la graphie de la semi-consonne finale.

À la ligne « bi » on constate en comparant le T1 et le T2 qu'il est possible d'arriver à des régressions lors du développement, c'est donc une très bonne idée de travailler avec un large nombre de syllabes pour pouvoir les détecter au plus tôt.

Structure du projet de code

Le projet F# **FsmParser** contient le code du compilateur. Il génère un programme en ligne de commande qui prend deux arguments : le chemin vers le fichier de graph à lire (.dot) et le chemin du fichier à utiliser pour la sortie. La sortie du programme, lorsque l'entrée a pu être interprétée correctement est un fichier C# qui contient une classe partielle[51] contenant les transitions de l'automate. Les transitions font appels à des méthodes partielles également générées par le compilateur dans ce fichier pour l'exécution du code lié à la production de l'analyse de sortie (remplissage d'une instance de class *Syllable*).

Ce fichier est utilisé par le projet C# **VietSyllableTransducer** dans un fichier nommé *VstParial.cs*. Le code effectif des méthodes partielles est implémenté si besoin est dans le fichier *VietSyllableTransducer.cs* ; de nombreuses transitions n'ont pas d'effet direct sur la structure de la syllabe et ne nécessitent donc pas l'ajout de code par le développeur.

Le projet **YieldMachine** est à l'origine un petit projet open-source en C# proposant une implémentation d'automates finis déterministes en (ab)usant du mot clef *yield* du langage. Il est utilisé en tant que bibliothèque de code par le projet VietSyllableTransducer.

Améliorations possibles

Comme la réalisation de cet automate de reconnaissance n'est pas le sujet principal de ce mémoire, des points complémentaires d'améliorations qui auraient pu être réalisés ne l'ont pas été afin de pouvoir consacrer le temps idoine aux autres éléments. Nous pouvons cependant en relever certains.

Premièrement il est à signaler que le chemin d'analyse des syllabes contenant le graphème « ê » n'est pas implémenté. La forme générale du chemin d'analyse ne diffère pas sensiblement de l'existant.

Ensuite, l'automate généré n'est n'a pas été prouvé comme étant minimal. Il pourrait être intéressant de faire fonctionner un algorithme de minimisation existant (algorithme de Brzozowski, de Moore ou de Hopcroft) afin d'en trouver une version minimale ou de démontrer que celle existante est minimale. Une réduction du nombre d'état, même petite, pourrait améliorer la lisibilité et la compréhensibilité de la représentation graphique de l'automate et donc sa vérification ou sa modification ultérieure.

Un algorithme de transformation de l'automate en expression régulière pourrait être appliqué à l'automate. De cette façon on obtiendrait une expression régulière permettant de reconnaître une

[50] Ce qui est linguistiquement faux, puisqu'il s'agit d'une semi-consonne. Heureusement, la fonction responsable de cette classification peut être modifiée.
[51] Cf. documentation du langage : https://msdn.microsoft.com/en-us/library/wa80x488.aspx

syllabe du vietnamien. Enfin, il pourrait être intéressant d'en produire une version produisant en sortie des phonèmes représentés par des symboles de l'alphabet phonétique international.

Conclusion

En l'état les outils de reconnaissance optique de caractères ne permettent pas de réaliser une reconnaissance « out of the box » d'un document mélangeant dans une même ligne différentes langues et différents systèmes d'écriture. En revanche, le logiciel open-source Tesseract permet grâce à son API de développer des programmes permettant de gérer ce fait.

Cette approche nécessite cependant des informations complémentaires sur le contenu des lignes en cours de reconnaissance. Ces informations peuvent être créées à partir de la reconnaissance du français qui est de relativement bonne qualité, et d'une connaissance préalable de la structure du texte à analyser. Cependant faire coller la théorie, c'est à dire grammaire qui décrit la structure du dictionnaire, avec la pratique (la sortie réelle d'une OCR) requiert, dans l'approche décrite dans ce travail, une tâche de classification pour laquelle des techniques de *machine learning* peuvent être mises à profit.

Diverses techniques, parfois relativement simples comme celle qui tire profit de l'ordre l'alphabétique qu'utilise un dictionnaire, permettent de reconnaître des erreurs dans la sortie d'une OCR. Si cette détection d'erreur est intégrée au programme de reconnaissance en tant que « boucle de feedback », il est alors possible d'utiliser cette connaissance pour retenter une reconnaissance.

Enfin, des techniques plus avancées peuvent être mises en œuvre (par exemple dans ce mémoire la comparaison avec des reconstructions du chinois moyen). Ces dernières ne nécessitent pas un haut niveau technique pour être implémentées, mais dépendent en revanche lourdement d'une bonne analyse linguistique en amont, et parfois de la disponibilité de données prêtes à l'emploi.

Bibliographie et références

Les URL données ici ont été vérifiées comme étant accessibles au moment de l'écriture. Cela peut cependant changer rapidement (deux liens ont changés durant la rédaction de mémoire) et elles ne sont donc données qu'à titre indicatif.

Articles

Karlgren, Bernhard (1922). The reconstruction of Ancient Chinese. *T'oung Pao*, vol. 21, pp. 1–42, 1922.
Tiré de http://www.archive.org/stream/s2tungpaotoungp21corduoft#page/n3/mode/2up

Maspero, Henri (1912). Études sur la phonétique historique de la langue annamite. Les initiales. *Bulletin de l'École française d'Extrême-Orient*. Tome 12, 1912. pp. 1-124.
Tiré de http://www.persee.fr/web/revues/home/prescript/article/befeo_0336-1519_1912_num_12_1_2713

Maspero, Henri (1920). Le dialecte de Tch'ang-ngan sous les T'ang. *Bulletin de l'École française d'Extrême-Orient*. Tome 20, 1920. pp. 1-119.
Tiré de http://www.persee.fr/web/revues/home/prescript/article/befeo_0336-1519_1920_num_20_1_5549

Maspero, Henri (1916). Études sur la phonologie chinoise. *Bulletin de l'École française d'Extrême-Orient*. Vol. 16, n° 1, pp. 61-73.
Tiré de http://www.persee.fr/doc/befeo_0336-1519_1916_num_16_1_5301

Nguyên, Phu Phong (1978). À propos du Nôm, écriture démotique vietnamienne. *Cahiers de linguistique - Asie orientale*, vol. 4 n°1, pp. 43-55.
Tiré de http://www.persee.fr/web/revues/home/prescript/article/clao_0153-3320_1978_num_4_1_1047

Livres

Hippisley, Andrew (2010). « Lexical Analysis » dans Indurkhya, Nitin et Damerau, Fred J. (dir.), *Handbook of Natural Language Processing Second Edition*, Boca Raton, Chapman & Hall/CRC.

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

Sagart, Laurent (1999). The Roots of Old Chinese. Amsterdam: John Benjamins.

Thèses et mémoires

Huynh, Mai Trang (2013). La conscience phonologique des enfants vietnamiens: son développement, ses liens avec la lecture et l'écriture et l'impact d'un entraînement précoce. (Thèse de doctorat, Université Libre de Bruxelles, Bruxelles, Belgique).
Tiré https://dipot.ulb.ac.be/dspace/bitstream/2013/209389/4/b7138b86-e939-4c0e-9215-b02a31f36648.txt

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

Lecailliez, Louis (2015). Méthode automatisée pour la recherche de caractères chinois complexes dans un dictionnaire électronique basée sur leur décomposition structurelle (version d'octobre). (Dossier pour l'obtention du Master 1 « Études Japonaises », Université Paris Diderot, Paris, France). Tiré de http://unixorn.azurewebsites.net/dl/dossier_Lecailliez_4bc14994-11c8-4dc5-a7cb-7094aeab3233.pdf

Nguyen, Quoc Cuong (2002). Reconnaissance de la parole en langue vietnamienne. (Thèse pour obtenir le grade de docteur de l'INPG, Institut national polytechnique de Grenoble, Grenoble, France). Tiré de www.afcp-parole.org/doc/theses/theseQN02.pdf.gz

Sites web

Kanji Datase Project (2015). 字形ＩＤＳデータ. (jikei IDS dēta)
Tiré de http://kanji-database.sourceforge.net/ids/ids.html

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

Source de données

Baxter, William H. et Sagart, Laurent (2014). Baxter-Sagart Old Chinese reconstruction, version 1.1 (20 September 2014).
Tiré de http://ocbaxtersagart.lsait.lsa.umich.edu/BaxterSagartOCbyMandarinMC2014-09-20.pdf

Nguyen, Trung (2012-2015). Chunom.org. Grade 1 Characters. Grade 2 Characters.
Tiré de http://www.chunom.org/

Annexe A : résultats d'OCR

Document 1 : OCR en français de la page 22 (0062)

——+:-(22)-e+——

vant à faire des ustensiles de ménage et des instruments de musique genre mandoline.

Bäo ÏÊÎK Violent orage, tempête avec pluie et grêle, ouragan.

Bäo un o fi, forte tempête. — Bâo lut oîfi, orage suivi d'inondation.—Bau bâo fi o, colique très dangereuse. — Chim bâo lhanh 5g

o î, autruche.

Brio Proteger, défendre; conserver, garantir, répondre de ; avertir, annoncer. Voir bàu.

Khuyëzi bào o, exhorter. —-

Dqg bào [lfE o, enseigner, conseiller. _

— Clic' bäo ÿâ o, Inontrer, indiquer. — Bào hg} o ä, protéger. — Ouan bäo hg) "È" o , Ibnctionnaire chargé d'un protectorat. — Bäo {in o fâ, donner des nouvelles, faire connaître un l'ait. -— Bâo länh o fi, cautionner. se rendre responsable. — Brio chüm O ÿ], tirer dalïaire, sauver d'une situation. — Bào trierîn' O ê, prévenir, avertir (l'avance.

Bâo ÏÏJK Embrasser, citreintlre; porter sur les bras, contenir.

J' Ë " w z

Bap Mais; chou. (Forme des S.A. mgîc 7k, arbre, et phù 'f, nom de plantes à épis de grandes dimensions.)

Cäi bäp ËE o. chou à cœur dur, chou pomme. ——— Bäp chiuii O fi, fleur de bananier. — eVzii lÿp biîp [lfi llj o, parler trop vite, manger les

mots. ——— Bäp cày o fil, manche de charrue. — Bäîp vê oflç, cuisse, lemur. — Bäp tay oÿfi, lavant-bras. — Bäp chmn o QË, la jambe. le tibia. lemollet.

Bal *. Prendre d'assaul; arra-cher, extirper, déraciner; être en-. traîné, être porté par le courant, sïêchouer.

Bat lhành o ÿfi, {emparer d'une citadelle. — Bat Igec o j], déployer toutes ses forces. — Tàu bru fig o, na\ire échoué. — Säng bot vào O Q1', le flot nous porte vers la rivage. — I/'ri bqt chqt fi o g, paroles vides

de sens, discours inutiles.

Bcit fit Prendre soin de; ouvrir, exclure; répandre, publier.

Brit ÊÂV. Écuelle, bol, plat.

Mot bât cœm {Q ofifl', un bol de riz cuit. — Truyèn y bât fi æ o, transmettre la robe et Yécuelle, c.-à-d. transmettre la place ou la l'onction à un successeur.

Bail 7Ê*. Séparer; dos à dos; op-

posés l'un à l'antre. Car. radical.

Bzit Â *. Huit(forme simple). Car. radical. A. V. Tribord.

Bât nghi o fi, les huit catégories de coupables qui ne peuvent être frappés que par sentence royale. ——— Bàt tiën o m], les huit immortels, les huit génies. — Bât dan o 'ê, les huit sons musicaux, un orchestre. —

Thdp bât + o, dix-huit.

Bai! m *. Huit(forme compliquée). A. V. Appuyer à droite, venir sur tribord; rejeter, repousser.

Bât quäi o ä», figure géométrique

la faute (*la* ou lieu de *le*) est dans le texte

employée pour la divinntion. (Cette
figure comporte huit divisions anx-

Document 2 : OCR en vietnamien de la page 22 (0062)

-*iF8'(ÎẼ)ơcd°-

\"ant ầl 1`aiI`e des Ilstensỉỉes de llìớể-
lìage et des inStl'llIIìents đe llllll-
Sỉ(I[10 genl'e nlanđoline.

Blĩo ẵ*. Vìolentn Ol'age, Í8lìIl)ÔÍ8
ấl\'oc pỉuìe et gl`èle, 01Il'agan.

BIỈO tI_It O Ẹ̃, Í`0l't.e ìellll)ểỉ8. -
Bão II_4t OĨẼ, orage Sui\'ì đ`ỉnonda-
t`ỉ0I1.-Đau bảo É O, colỉque tl'ì*S
dangerellse. - Cltiln bão tllanll ,gg

O Ế, alltrllclìe.

Btìo l)l'(ìívÊẾg8l`, đéfen(ll`e; con-
S<`I`\'eI`, gal'antll`, I`t'*l)<›ndl't' (le ; a\7ol'-
tỉl', ẽlnnoncel'. \"i0Íl' btìlt.

Ắylllllgểìl blì0 O, Pxll0l`ter. -

Dạ_q l›ả0 ỊlỈỄ O, 0nS('ĨgH0l', conseillor. _

- Cllễ bảo ỈỀ O, Inontrel', indítlllel'.
- Btầo llfì O Ế, l)l'Otlígor. - Ọllan
btìo lig) “Ê” O , íbIlctỉ0IIllail'e fllargé
(l”1ln I)r0tect0l'at. - B(ỉn tin o fẵ,
d(›nnol' dPS n01l\"elleS, faire connaĩtre
llll fait. - Btìo lãnh O Ế, rílllltỉon-
llel'. So l'ell(ll'(' l'oSp0l1sabl0. – Bào
cìIIỈ›a O EJ, lỉl'el' đ`aÍỈ`aìI`í', Sau\'0l'
(l'une Situation. - Báo trllìóư' O Ẹ,
pré\'ollìI', av"el'[ỉl` «l`a\`alì('<*.

Bảo ĨẾJ*. I*ÌllìỈ)l`asSol“, c"Il'ein(Ìl'e;
1)0l'Íel' Slll` les l)raS, c0ntenỉl`.

Ổ Ế " ư I

Bup Mais; ('.llO1I. (l'Ol'IIìe des S.aÀ.
Iltộc ỹk, al'l›re, etn phlì “JỀ, nom de
planl,eS ìl él›iS de gl'andeS dinlensìons.)
Cfỉi biìp Ếĩ O. Clloll ằl ('(Pur dtlr,
clloll Ịìolllnìế. -- Bắp clclclỉi O Ẹ̃,

íleIll' (le banalỉ0l`. - ,Vlíi lgĩp bắp [lệ] IIỀ O, lJãll'leI' I.I'0l) \`ỉte, nìếlnger les nlots. - Bắp cI`l_Ij O ả¡, nìallclle de Cllarl`ue. - Bìỉp uế OẸ¡, cuỉsse, íềnìllr. - Bắp tay oặ¡, l`a\'ant- l›raS. - Bắp cì!o›7l O iã, Ỉa janìb1`. IP tỉbỉa. le°mOllet.

Bạt *. Pl'endre đ°aSSaIlt; arra- (fllel', PXỈ›il`lì8l', dếl'aciner; être en-- Il'aìl1é, êtl'e porté par le c0lll'ant, S,ỐCll()ll8l'.

Bạt II0.l`1lÌl O bì, s'emparer (l'une ('ìtãI(lelle. - Bạt lf_l'L' O jj, déployer tolltes SPS forces. - Tàu bạt gg O, lìa\ỉre éclloué. - Sóllg bạt vào O gị, le Ỉlot nous porle \fers la rivlge. - Ìl`»í bạt chạt Ế O jg, paroles vỉdes

(le Sens, díscollrs ỉnutỉles.

Bát Ệ*. Pl`endre Soỉn đe; oIl\'l'ỉl', excllre; I'épandre, publier.

Bát ỀĂĨ*. Ếctlelle, bol, I)laí.

Mọt bát cưm ĩĩ oiữ, nn bol de ríz cuìt. - Tmgền _tl bát ị Ấ O, transmettre la robe Pt l`ẻcuelle, 0.-à-d. transmettre la plarọ Oll la lbnctỉon à un Successeur.

Bát 7Ý*. Séparer; dos à dos; op-

posés l`Iln à l°alltre. Cal'. l'ađỉcal.

Bát Ã *. Hllỉt(fOl'me sìnlple). Car. radical. A. V. Trỉbord.

Bát nghi O ị, les lluit catégories de coupables quỉ ne peuvent être frappés que par sentence toyale. - Bài tiết: O 1ị][, les lluỉt ỉmlnorlaels, les lluỉt génỉPs. - Bát âlll O Ệ, ỉes lluit Sons nlusiraux, un orrllestre. -

Thập bát -I- O, dix-llllỉt..

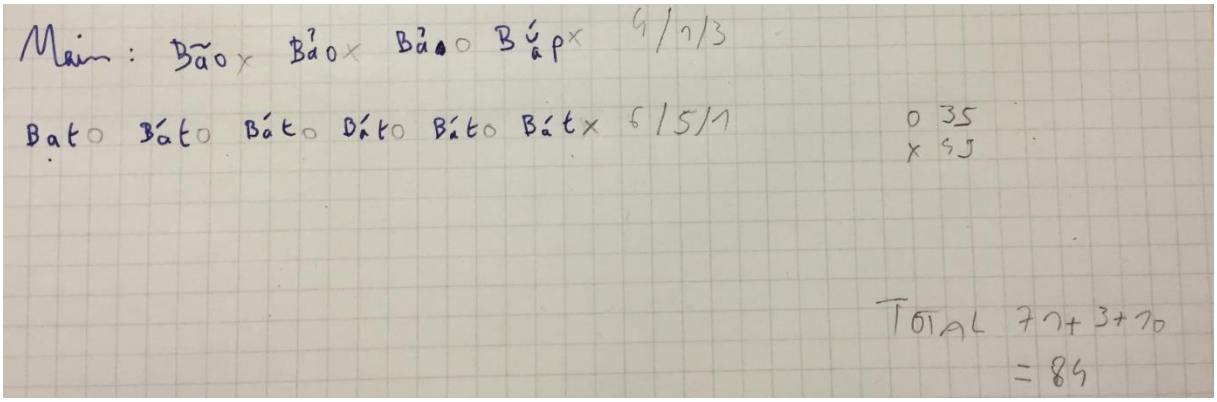
Blít W *. Hllỉt(f0l'nle colnpỉỉquée).
A. V. Appuyel' à droỉte, venỉr sur
lrỉbord; rejeter, repollsser.

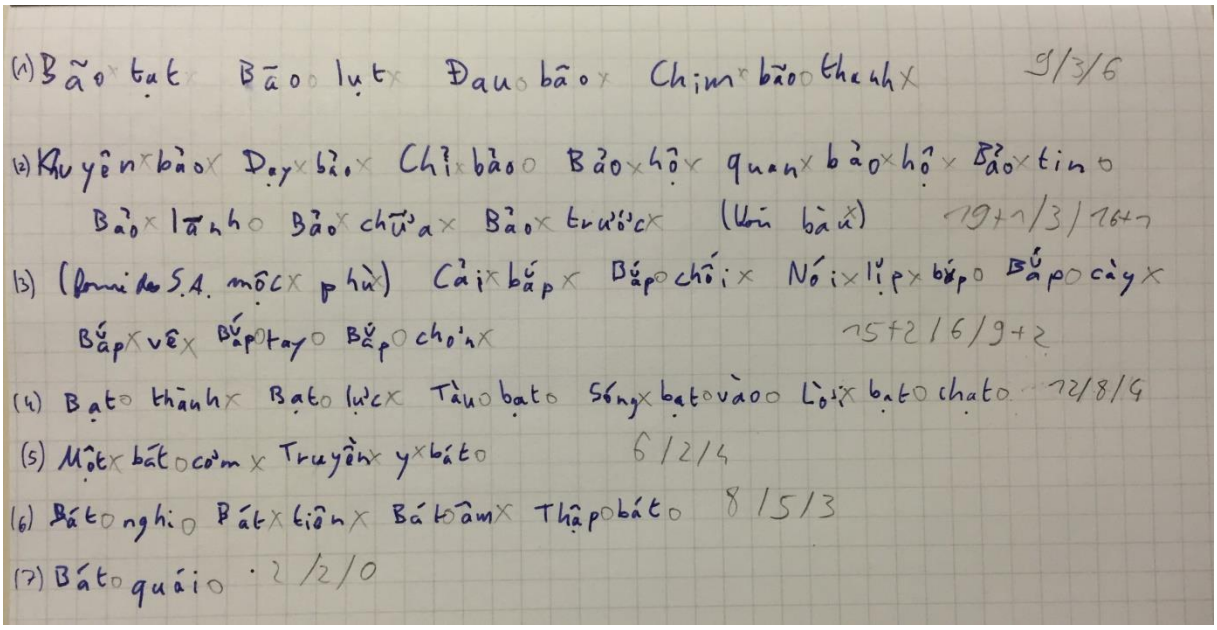
Bát quái O ẩ`, figure géométriqie
employée pour la divinltỉnn. .(Cette
figure Pomporte lluỉt dìvỉlinns anx-

Document 3 : décompte des syllabes correctes du document 2

Un rond indique que la syllabe es correctement reconnue, une croix qu'elle ne l'a pas été.

Main regroupe les syllabes de tête d'entrée. Le reste des syllabes est dans le même ordre d'apparition que dans le document d'origine.





Document 4 : OCR en chinois de la page 197 (0237)

轟躍擊伊薈荁，莘 鷲蠹“ 蔽 O 艫, desceudre 芝 i b0rd
(l'un navire' s 弔 enlba】¯(l[【er. 一 (宴， (芝“'(差，玨
`g'′】鬱 於 O 屢箕， demeurer danS llIle
l)ar(lue. 一 麗荁 (荁， (互帕荔，玨 几 於 O ' cellx

(l'l】i s0nt ell desS0uS' 丑 eS inf(圭 rieurs.

三差二 〣' 〣J， 叉 容. T0l(巨 rel" laiSSel` fail 〣e,

_貘 .囊 i|王 ser (i'in〔lll 丑 gence; asPect' figul'e'
暫′ 舞 0 實 me. 〈c0!`ruPti0【l du S. A. 乙玨@′ 万漥,

亡 〣i】】嘍 me car.' m 巷 me Signi 蠶 cati0n.〉

丑觼(窒濟 O 竇實差一 姍′粵槐仍 O 饒, se Pard0nner,
Se r(巨 c0llciiiel¯. 一 丑!′沸互窪 佩′》觼獅 O 顏,
l)eaut(圭 deS f0rmes' aSPect ext(圭 l'ieul¯.
一一 P′l 鈸 G r 丑狒籔荁觼'畜 芙 O， esP 馨 ce d'蓼`])re
盅 c0t0n.

，¯!亡「.._v 〣 〣′溯. Eau qui (蠹 0ll 丑 e; Suivl`e
荁′ 啻霾 c0urant (ne S'emPl0ie (壬 u'en
二謯 0mP0Siti0n). v0ir 茲茲獅窪. (F0rm(巨 des
差¯， S. A_ z′l2 躉婁' 女 l(， eau， et ′五 t 罩 ntl 賓 用， user.)
丑_贗′互，乙漥 盅(宴 O 捕， rejeter' reP0uSser,
鷲 efuser' 豐 e d(巨 ba【¯raSSer (le' 一 C 窿差
茲觼茲嚐 傲 O ' n0m de P0iSSOn.

.′ ¯，.′′髀昜*. SPlendeul、 dll (€ie 丑;
¯妻差斷荁蠹 rt(巨 ' 丑 umi 蠢 re; l)rinciPe m 蠢 le ' Pal 〣
_¯薈 l 襲 l)0Siti0n 盅 d 茲 m 陰' (lui eSt]e
蠹 l`inciPe femelle; 】nat 丑 l 蔓三 re P【l】¯e;
′， i`en' h0n' beau; j0ur' cie 丑' S0leii.

–¯ 丑_素佣聾浬漥鋤娜 O 問 ' le nl0nde n]0ra 丑.
〣躉 一一 z 乩 m 茲屾躪澧一 陰 O , 丑 es deux grands

¯P 鸞 i 皿 ciPes 0PP0s 巨 s (c 丑 art(巨 et t(巨 n 屢三 l)reS'
皿 llit et j0ur, l)ien et ma】， nl 蠹董 le e|二
吽 feme]ie)， 一 K 龜筆 j 淜″狐 g 氣 O， air
'}〈鼻¯¯薹)ul" Parties (巨 thG 差 r(巨 eS et subti 丑 eS de
`′ 〣 〣蠟 m 魑 ti 醫 re' “uide P0sitif. 一 丑觼)0 刃)霾
' . ′傷鷹丑畫豚粵贅_壩豚江 O 起 石， amiante.一 D 恢伊 n 漥

臟 . .
揚*. Exclter' s0uleVer' agl-

–” 〦9.(197).€垚.一一

teI¯, I'遙 I)andre all 丑 0in' (巨 ten(ll`e la
r 茗堅 PuI]atiOn , Pul)lieI" rendre c(芒 l(巨 hre.

刀´´，0，”臏 乙 Zrz 刃´乙 O 名' 一 Puh]ier un
n0lll' c(蔓丑巨 breI¯ la I¯en0mnl 鱈 e de (Iue 丑一
qu'un_ 一一 曠尸酚觼琍窪 乙丑岫@'苑蟲尸 腳 嬤(r 差尸 0(I 玨 張
O 於 夕卜' 玨二′ I 互 re c0nna 竇 tre au deh0rS
ce (Iui Se PaSSe dedanS' diVu 丑 guer
leS SecretS de fami 丑丑 e. 一 Z 伊互芝拻乙′ 乙 玆!腴 0，” 蠶
明 O' d 差】]l0n【rer' ex【)0Ser' reI】dre
II]an 丑 feste.

丑岔鬱'0，獅濫 才昜*. Pin deS Pag0deS， cY 一
l)I`亡 S， l〕euP 丑 ieI、; n0II] (]e fam 丑 ile.

C 么途婁´´乙”宏”´z´Z“，0w'r 稟 瘠亥 黃 O' buis de
Chine. 一一 窒 〦´乙槐儷′矗´丑薹汔臾@，7 乙澧尸 青 O' 丑'arbre
t0uj0lll`S vert' 丑 e S 戛 Iu 丑 e.

臟乙鬱'伊，乙濫 羊*. Ch(荃 Vl`e'ln0ut0n. Gar.

radic…l 丑. v0iI¯ (箸(窒 et c´鬱鈸(宴獅.

`s′ 0'7 乙 瀕鶖夕@懈 山 O' c 玨 leVreui 丑， cha 一
ll]0is. 一 (窒(聶犖 S0'n 鳳娜，0，咭 葷鬼 山 O'
c0rne deberger. 一 丑眈'0 咧浬 Z 憊盅，!´O 酒'
nl0llt0Il et vin (Pr(巨 Sent de n0ceS).

丑乜矗，伊姍霾' 手羊*. AIIeI' de t0lls C6t(圭 S
sanS l)llt d 鱈 termin(毛' c0IlriI`(〦二芝 I et
i 矗， eI`rer 矗 Sa fantaiSie.

丑祕'0'昭 曼羊*. 0c(巨 an，、´aSte】ller;(Ill 丑

S'(巨 tend au l0in; (巨 trangeI`， ext(巨 I¯ielll'.

臟鬱狎@啼魑也賓晝'´乙(荁玨 O 薰每' I'inln]enSit(圭 d(*S
n]erS， 丑 eS vaSteS 0c(圭 anS. 一一 丑(荁佩矗!'
乙丑乙粵'0 鞭互'r 窐 束 O， leS nlerS 0rienta 丑 es
(丑'Ind0 一 Chine fran(三 aiSe). 一 Zv´窒鍍哩玨

鋮“，0 噌 夕卜 O' n]eI'S ext(巨 rieureS' Pays
I0 丑 ntainS. 一 窒泫〝(差姍薈窒〝〔玆鱈獅豊一 玆鋮噌 玆輦乙獅獅澧一
璁 系充 束 O , titl'e (圭 c0【】r【(牟 (丑 u g0llver 一
nellrg(巨 n 夤巨 raide i'Ind0-cI】ine 玨`r 乏!n(差 a 啻 Se.
一一 蠶轟互 攪彶觼噌 悔 O, 丑 e n0nl d'I】ne

¯Province dll 'l'0n|(in (l).

Document 5 : résultat de l'analyse de l'agencement de la page 22

vant à faire des ustensiles de ménage et des instruments de musique genre mandoline.

Bão 雹*. Violent orage, tempête avec pluie et grêle, ouragan.

Bão tạt ○ 悉, forte tempête. — *Bão lụt* ○ 潵, orage suivi d'inondation. — *Đau bão* 疱 ○, colique très dangereuse. — *Chim bão thanh* 鴇 ○ 青, autruche.

Bảo 保*. Protéger, défendre; conserver, garantir, répondre de; avertir, annoncer. Voir *bàu*.

Khuyên bảo 勸 ○, exhorter. — *Dạy bảo* 吠 ○, enseigner, conseiller. — *Chỉ bảo* 指 ○, montrer, indiquer. — *Bảo hộ* ○ 護, protéger. — *Quan bảo hộ* 官 ○ 護, fonctionnaire chargé d'un protectorat. — *Bảo tin* ○ 信, donner des nouvelles, faire connaître un fait. — *Bảo lãnh* ○ 領, cautionner, se rendre responsable. — *Bảo chữa* ○ 助, tirer d'affaire, sauver d'une situation. — *Bảo trước* ○ 畧, prévenir, avertir d'avance.

Bảo 抱*. Embrasser, étreindre; porter sur les bras, contenir.

Bắp 粽. Maïs; chou. (Formé des S. A. *mộc* 木, arbre, et *phù* 芣, nom de plantes à épis de grandes dimensions.)

Cải bắp 菠 ○, chou à cœur dur, chou pommé. — *Bắp chuối* ○ 桂, fleur de bananier. — *Nói lịp bắp* 吶 啦 ○, parler trop vite, manger les mots. — *Bắp cày* ○ 耜, manche de charrue. — *Bắp vế* ○ 髀, cuisse, fémur. — *Bắp tay* ○ 捭, l'avant-bras. — *Bắp chơn* ○ 蹎, la jambe, le tibia, le mollet.

Bạt 拔*. Prendre d'assaut; arracher, extirper, déraciner; être entraîné, être porté par le courant, s'échouer.

Bạt thành ○ 城, s'emparer d'une citadelle. — *Bạt lực* ○ 力, déployer toutes ses forces. — *Tàu bạt* 艚 ○, navire échoué. — *Sóng bạt vào* 涛 ○ 𠬜, le flot nous porte vers la rivage. — *Lời bạt chạt* 衕 ○ 撦, paroles vides de sens, discours inutiles.

Bát 撥*. Prendre soin de; ouvrir, exclure; répandre, publier.

Bát 鉢*. Écuelle, bol, plat.

Một bát cơm 沒 ○ 飣, un bol de riz cuit. — *Truyền y bát* 傳 衣 ○, transmettre la robe et l'écuelle, c.-à-d. transmettre la place ou la fonction à un successeur.

Bát 丷*. Séparer; dos à dos; opposés l'un à l'autre. Car. radical.

Bát 八*. Huit (forme simple). Car. radical. A. V. Tribord.

Bát nghi ○ 儀, les huit catégories de coupables qui ne peuvent être frappés que par sentence royale. — *Bát tiên* ○ 仙, les huit immortels, les huit génies. — *Bát âm* ○ 音, les huit sons musicaux, un orchestre. — *Thập bát* 十 ○, dix-huit.

Bát 捌*. Huit (forme compliquée). A. V. Appuyer à droite, venir sur tribord; rejeter, repousser.

Bát quái ○ 卦, figure géométrique employée pour la divination. (Cette figure comporte huit divisions aux-

Annexe B : code de programmes mineurs

Document 1 : écriture du résultat de « boxing » dans un fichier texte (C++)

```
#include <tesseract/baseapi.h>
#include <leptonica/allheaders.h>

int main() {
   Pix *image =
pixRead("/Users/Louis/Downloads/dictionnaireanna01bone_jp2/test/dictionnaireanna01bone_0062.
jpg");
tesseract::TessBaseAPI *api = new tesseract::TessBaseAPI();
 api->Init(NULL, "eng");
 api->SetImage(image);
 Boxa* boxes = api->GetComponentImages(tesseract::RIL_TEXTLINE, true, NULL, NULL);
 printf("Found %d textline image components.\n", boxes->n);
 for (int i = 0; i < boxes->n; i++) {
   BOX* box = boxaGetBox(boxes, i, L_CLONE);
   api->SetRectangle(box->x, box->y, box->w, box->h);
   char* ocrResult = api->GetUTF8Text();
   int conf = api->MeanTextConf();
   fprintf(stdout, "%d;%d;%d;%d\n", box->x, box->y, box->w, box->h);
 }
}
```

Document 2 : programme C# de dessin des « boîtes » générées par Tesseract

```
using System;
using System.IO;
using System.Drawing;
using System.Collections.Generic;
using System.Text.RegularExpressions;

public class Program {
	public static void Main(string[] args) {
		if(args.Length < 2) {
			Console.Error.WriteLine("Nombre d'arguments insuffisants.");
			Console.Error.WriteLine("usage : draw_rect.exe <image> <box_file>");
			Environment.Exit(1);

			if(!args[0].Contains(".jpg")) {
				Console.Error.WriteLine("Nom de fichier d'image incorrect.
Fichier .jpg attendu.");
			}
		}
		Console.WriteLine("Loading image...");
		Image img = Image.FromFile(args[0]);
		Console.WriteLine("Image loaded.");
		Pen pen = new Pen(Color.Red, 1);
		int k = 0;
		Regex regex = new Regex(@"^\d+;\d+;\d+;\d+$");
		using(StreamReader sr = new StreamReader(args[1])) {
			Graphics g = Graphics.FromImage(img);
```

```
                        while(!sr.EndOfStream) {
                                string line = sr.ReadLine();
                                if(!regex.IsMatch(line)) {
                                        continue;
                                }
                                List<int> num = new List<int>();
                                string[] values = line.Split(';');
                                for(int i = 0; i < 4; i++) {
                                //Console.WriteLine(String.Format("string number [{0}]", values[i]));
                                        num.Add(Int32.Parse(values[i]));
                                }
                                Rectangle rect = new Rectangle(num[0], num[1], num[2], num[3]);
                                g.DrawRectangle(pen, rect);
                        }
                        g.Dispose();
                }
                img.Save(OutFilename(args[0]));
        }

        public static string OutFilename(string base_name) {
                string first_part = base_name.Substring(0, base_name.IndexOf(".jpg"));
                first_part += "_boxed.jpg";
                return first_part;
        }
}
```

Annexe C : Outils utilisés

Logiciels

Visual Studio

Environnement de développement pour Windows permettant de créer, compiler et débuguer des programmes écrit en C# et en F# (d'autres langages et plateformes cibles sont supportées).
Téléchargement : https://www.visualstudio.com/en-us/products/visual-studio-community-vs.aspx

Xamarin Studio

Environnement de développement pour OS X ou Windows permettant de créer compiler et débuguer des programmes écrit en C# et en F# et des applications mobiles multi-plateformes.
Téléchargement : https://xamarin.com/download

Tesseract

Logiciel open-source d'OCR supportant 39 langages « out of the box ».
Téléchargement : https://github.com/tesseract-ocr/tesseract
Exemple d'utilisation de l'API : https://code.google.com/p/tesseract-ocr/wiki/APIExample

Graphviz

Logiciel de visualisation de graphs.
Téléchargement : "Tesseract (custom)"

Bibliothèques

FsLex et FsYacc

Package qui fournit une implémentation de Lex et Yacc pour le langage F#.
Téléchargement : http://fsprojects.github.io/FsLexYacc/

YieldMachine

Bibliothèque et projet de démonstration pour la création d'automates finis déterministes en C#.
Téléchargement : https://github.com/eteeselink/YieldMachine

Programmes écrits

Analyse d'une syllabe vietnamienne & expériencesgithub.com/titanix/m2_vietnamese_analysis

Police de caractères

UNICODE Han Nom Font Set

Téléchargement : http://vietunicode.sourceforge.net/fonts/fonts_hannom.html

Hanazono

Téléchargement : http://fonts.jp/hanazono/