ÉCOLE NATIONALE DES CHARTES
UNIVERSITÉ PARIS, SCIENCES & LETTRES

Kelly Christensen
licenciée ès enseignement musical
diplômée de master musicologie
diplômée de doctorat musicologie

Modélisation des transcriptions ALTO avec la TEI

En complétant le pipeline du projet *Gallic(orpor)a*

Mémoire pour le diplôme de master
« Technologies numériques appliquées à l'histoire »

2022

Résumé

Quand des modèles OCR et HTR extraient les données d'une ressource textuelle numérisée, les informations relatives à la structure physique de l'image risquent de se perdre. Un schéma XML standardisé qui s'appelle ALTO a été créé afin de conserver et structurer ces données non-textuelles et géometriques en les tenant en relation avec le contenu textuel. La plupart des modèles OCR et HTR compte sur ce schéma. Cependant ALTO ne convient pas bien à l'édition numérique ni aux traitements automatique du langage. Les éditeurs et les chercheurs en lettres attendent un schéma XML plus courant dans le monde des humanités numériques : la TEI. Il faut donc un mapping pour transformer un fichier ALTO en fichier TEI sans perdre aucune donnée lors du processus. Cette transformation automatisée permet à conserver les données particulières au schéma ALTO, telles que celles sur la segmentation et sur la structure physique du document numérisé, ainsi qu'à exploiter le contenu textuel de la ressource textuelle. La flexibilité de la TEI et son usage très répandu rendent le schéma idéal pour mieux valoriser les données produites par les modèles OCR et HTR.

Dans le cadre du stage pour obtenir le diplôme de Master 2 « Technologies numériques appliquées à l'histoire », ce mémoire porte sur la modélisation de la transformation de ALTO en TEI. Cette modélisation a été réalisée dans le cadre du projet *Gallic(orpor)a*, financé par la BnF lors d'un stage qui a eu lieu au sein du laboratoire Automatic Language Modelling and Analysis & Computational Humanities entre avril et juillet 2022.

Mots-clés : HTR, OCR, ALTO, TEI, TAL, édition numérique.

Informations bibliographiques : Kelly Christensen, *Modélisation des transcriptions ALTO avec la TEI. En complétant le pipeline du projet* Gallic(orpor)a, mémoire de master « Technologies numériques appliquées à l'histoire », dir. [Noms des directeurs], École nationale des chartes, 2022.

Remerciements

Mes remerciements vont tout d'abord à...

Introduction

Première partie

Présentation du projet

Chapitre 1

Qu'est-ce que l'HTR ?

Qu'est-ce qu'est l'*Handwritten Text Recognition* ou, traduit en français, la reconnaissance du texte écrit à la main ? L'*Handwritten Text Recognition* (HTR) est l'un des meilleurs approches aujourd'hui à prédire du texte à partir d'une image numérique. Pour expliquer comment l'HTR se réalise, il faut exposer en premier temps qu'est-ce qu'une image numérique. En sachant à quoi ressemble la saisie d'un logiciel HTR, qui fait la prédiction du texte sur l'image, on comprend mieux le défi dont s'occupaient des chercheurs pendant près d'un siècle.

1.1 Les objectifs de l'HTR

1.1.1 Le contour

L'un des objectifs de l'HTR est d'imiter l'œil humain et reconnaître les lignes et les points d'une écriture sur une image numérisée du texte. Une image numérique se compose d'un quadrillage des carrés qui s'appellent des pixels. Venant de l'anglais, le mot pixel veut dire *picture element* et il décrit l'élément le plus minimal d'une image numérique. Les pixels sont stockés dans un format *bitmap* ou BMP et chaque pixel peut compter 1 bit, 4 bits, 8 bits, ou 24 bits selon l'encodage de l'image. Quelque soit l'encodage, chaque pixel n'a qu'une seule couleur. Vu ensemble, un groupe de pixels peut donner l'impression des courbes d'un objet ou d'un caractère. L'exemple de figure 1.1 pourrait donc montrer la diagonale de la lettre « A » écrite en crayon rouge.

Un scanner encode l'image en décomposant une image en les unités de pixel et en donnant à chaque pixel un tableau de trois entiers. (cf. Figure 1.2) Le premier entier qui se trouve dans la donnée tripartite d'un pixel déclare le degré de la couleur rouge à rendre dans la carré. Le deuxième entier déclare le degré de la couleur vert et le troisième de la couleur bleu. En visionnant de loin un groupe de ces carrés, l'œil humain arrive à distinguer les formes. L'HTR vise à reproduire le même résultat et reconnaître les points contiguës d'un caractère.

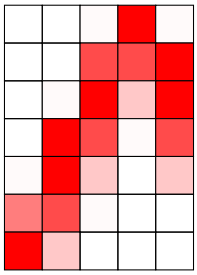


FIGURE 1.1 – Une couleur per pixel

255,255,255	255,255,255	255,250,250	255,0,0	255,250,250
255,255,255	255,255,255	255,75,75	255,75,75	255,0,0
255,255,255	255,250,250	255,0,0	255,200,200	255,0,0
255,255,255	255,0,0	255,75,75	255,250,250	255,75,75
255,250,250	255,0,0	255,200,200	255,255,255	255,200,200
255,125,125	255,75,75	255,250,250	255,255,255	255,255,255
255,0,0	255,200,200	255,255,255	255,255,255	255,255,255

FIGURE 1.2 – Donnée tripartite portant sur le degré du rouge, du vert, et du bleu

1.1.2 Le masque

L'autre objectif de l'HTR est de distinguer les limites et regrouper les composants d'un caractère. Même si un modèle HTR arrivait à reconnaître toutes les lignes de la lettre *A*, il n'arriverait pas pourtant à reconnaître que les lignes se constituent un caractère sans les rassembler et classifier comme un objet contiguë. Il faut donc tracer la limite d'un caractère dans une entité qui s'appelle un « masque ». [1]

1. Denis COQUENET, Clément CHATELAIN et Thierry PAQUET. « Handwritten Text Recognition : From Isolated Text Lines to Whole Documents ». In : *ORASIS 2021*. Saint Ferréol, France : Centre

1.1.3 Allier les deux objectifs

Un modèle HTR doit donc réaliser ces deux objectifs : identifier les contours d'un caractère et se rende compte que ces points contiguës constituent un objet contiguë. Cet objet contiguë permet au modèle reconnaître du texte. La reconnaissance du texte n'est pas de magie. Elle compte sur la constitution des objets ou des *masques* à partir des pixels. Le modèle HTR traite ces objets comme des matrices, qui s'expliquera prochainement, et peut donc réaliser des calculs qui lui permet prédire du texte.

1.2 Les origines de l'HTR

L'*Handwritten Text Recognition* a évolué à partir de l'*Optical Character Recognition*. Les origines de cette dernière méthode atteignent jusqu'au dix-neuvième siècle. Mais les progrès les plus importants à la technologie éventuelle dont le projet *Gallic(orpor)a* profite ont eu lieu il y a plus qu'un demi-siècle aux États-Unis. Le développement des logiciels OCR a été réalisé dans le cadre de l'amélioration du traitement en masse des données numérisées. Dans les années soixante, ce besoin de gérer et traiter des quantités vastes de données s'est fait senti largement dans les grandes sociétés, surtout les banques. Mais la numérisation des journaux et le traitement en masse des lettres à la poste étaient aussi quelqu'uns des contextes importants du développement de la reconnaissance du texte sur l'image.

Dans les années mille neuf cent soixante, plus en plus de sociétés voulaient adopter le traitement des données assisté par ordinateur. Ce traitement a exigé l'encodage des données dans un format directement exploitable par ordinateur. En gros il y avait deux moyens de la saisie des données. En premier temps, il y avait la carte perforée dans laquelle une machine ferait des trous qu'un ordinateur savait lire. En deuxième temps, il y avait la bande magnétique sur laquelle une machine imprimerait des bytes, la plus petite unité exploitable par ordinateur.

L'*Optical Character Recognition* comptait sur ce dernier moyen ainsi qu'une autre technologie, la numérisation des images par scanner. L'atout de l'OCR est qu'en principe sa saisie de données se fait automatiquement. À l'époque, les autres moyens d'encoder les données textuelles exigeaient la saisie manuelle des données. Dans le cas des cartes perforées, un individu lirait un document et taperait son texte à la perforatrice à clavier. Le *Magnetic Tape/Selectric Typewriter* (MT/ST), développé par l'entreprise américaine IBM en 1964, comptait sur la même saisie manuelle des données mais il les a encodé sur une bande magnétique au lieu d'une carte perforée.

En 1971, le chercheur Ben R. Schneider a comparé l'OCR et ces deux autres moyens

National de la Recherche Scientifique [CNRS], sept. 2021. URL : `https://hal.archives-ouvertes.fr/hal-03339648` (visité le 04/08/2022).

d'encodage du texte. [2] Schneider a déterminé que l'OCR n'avait pas encore surpassé l'appareil MT/ST d'IBM parce que, malgré sa saisie automatique de données, il exigeait toujours beaucoup de correction à la main. Contrairement aux deux autres méthodes, puisque les données d'entrée du traitement OCR n'étaient pas créées par un humain mais par un scanner, l'OCR n'a pas permis de la correction lors de la saisie. L'appareil MT/ST avait donc une exactitude supérieure à l'OCR à l'époque.

En outre, un logiciel OCR était limité par son jeu de données de polices, puisqu'il prédit du texte en faisant un comparaison entre un caractère qu'il a mémorisé et le motif qu'il a reconnu sur l'image. Même aujourd'hui l'OCR se distingue de son successeur l'HTR par le fait qu'il compte sur une base de données des polices. Contraire à l'OCR dans les années soixante-dix, l'appareil MT/ST pourrait encoder des documents des diverses polices en comptant sur le discernement d'un être humain lors de la saisie des données.

En 1977, ayant vu le progrès de la technologie, Gian Piero Zarri a résumé l'état de l'OCR. Il a remarqué que la reconnaissance du texte sur les manuscrits n'était pas encore faisable.

> Rappelons que la reconnaissance optique des caractères permet la lecture directe du texte par un « scanner » qui se charge d'effectuer le transfert sur bande magnétique ; le texte doit être composé avec des caractères de type « imprimerie » ou « machine à écrire », car la lecture de caractères « manuels » ne semble pas encore actuellement complètement sortie de la phase expérimentale. [3]

Il y a deux ans avant l'appréciation de Zarri, le chercheur américain Ray Kurzweil a produit son lecture *Kurzweil* ou le *Kurzweil Reading Machine* (KRM). L'appareil a avancé l'OCR en pouvant reconnaître plusieurs polices. [4] Cependant, comme dit Zarri, l'OCR restait sous la dépendance de la reconnaissance des polices et donc ne pouvait pas encore parvenir à la prédiction du texte écrit, ce qui a poussé le développement de l'*Handwritten Text Recognition*.

1.2.1 La binarisation

Afin de reconnaître la police d'un caractère, les logiciels OCR du XXe siècle avaient besoin d'un processus préliminaire qui s'appelle la binarisation. Cette étape est toujours

2. Ben R. Schneider. « The Production of Machine-Readable Text : Some of the Variables ». In : *Computers and the Humanities* 6.1 (sept. 1971), p. 39-47. ISSN : 0010-4817, 1572-8412. DOI : 10.1007/BF02402324. URL : http://link.springer.com/10.1007/BF02402324 (visité le 02/08/2022).

3. Gian Piero Zarri. « Quelques aspects techniques de l'exploitation informatique des documents textuels : saisie des données et problèmes de sortie ». In : *Publications de l'École Française de Rome* 31.1 (1977), p. 399-413. URL : https://www.persee.fr/doc/efr_0000-0000_1977_act_31_1_2286 (visité le 01/08/2022).

4. Gregory Goodrich. « Kurzweil Reading Machine : A Partial Evaluation of Its Optical Character Recognition Error Rate ». In : *Journal of Visual Impairment and Blindness* (12 jan. 1979).

nécessaire pour l'HTR puisque le logiciel, quelque soit sa manière de reconnaître des motifs dans une image, a besoin de relever d'un tableau de pixels les contours d'un caractère. La binarisation trie les pixels d'une image en deux classes : l'arrière-plan (*background* en anglais) et le premier plan (*foreground* en anglais).

Le niveau de gris

Normalement pour binariser une image, on veut remplacer la valeur composée d'un pixel, c'est-à-dire les trois degrés du rouge, du vert, et du bleu, avec la valeur simple d'un décimal. Ce décimal veut représenter l'échelle des gris dans le pixel. En anglais, ce traitement s'appelle le *grayscale* ou le niveau de gris en français. Aujourd'hui les langages de programmation ont souvent des librairies qui fournissent des méthodes pour rendre une image en niveau de gris automatiquement. Mais dans la Figure 1.3, nous donnons un calcul simple qui sert à montrer en exemple le traitement niveau de gris. On commence toujours avec la donnée tripartite du pixel, qu'on veut remplacer par une seule valeur. Représentons chaque partie de la donnée du pixel par les variables p :

$$p_1, p_2, p_3$$

En premier temps, on prend la moyenne des degrés de la couleur, c'est-à-dire la moyenne de p ou $\bar{p}$. La Figure 1.5a montre le résultat de ce calcul superposé à l'image originale.

$$\bar{p} = \sum_{i=1}^{3} \frac{1}{3} p_i = \frac{1}{3}(p_1 + p_2 + p_3)$$

Ensuite, on récupère $\bar{p}$ et la divise par la valeur maximale du p, qui est 255 parce que le degré du rouge, vert, ou bleu d'un pixel ne monte qu'à 255. La Figure 1.5b montre le résultat de ce calcul.

Donnée tripartite du pixel p_1, p_2, p_3	Moyenne du pixel $\sum_{i=1}^{3} p_i$	Valeur niveau de gris du pixel $\frac{\sum_{i=1}^{3} p_i}{\max\{p_i\}}$
255,000,000	$\bar{p} = \frac{255+0+0}{3} = 85$	$\frac{\bar{p}}{255} = 0.33$
255,075,075	$\bar{p} = \frac{255+75+75}{3} = 135$	$\frac{\bar{p}}{255} = 0.53$
255,125,125	$\bar{p} = \frac{255+125+125}{3} = 168.33$	$\frac{\bar{p}}{255} = 0.66$
255,200,200	$\bar{p} = \frac{255+200+200}{3} = 218.33$	$\frac{\bar{p}}{255} = 0.86$
255,250,250	$\bar{p} = \frac{255+220+220}{3} = 251.67$	$\frac{\bar{p}}{255} = 0.99$
255,255,255	$\bar{p} = \frac{255+255+255}{3} = 255$	$\frac{\bar{p}}{255} = 1.00$

FIGURE 1.3 – Évaluation des pixels

Le seuil

Il y a plusieurs techniques de binarisation mais toute a besoin d'un seuil (*threshold* en anglais). Le seuil nous permet à trier les pixels en les deux classes : le *background* et le *foreground*. Nos yeux font cette étape facilement, mais un ordinateur a besoin d'un algorithme. L'un des algorithmes le plus courant pour définir le seuil a été élaboré en 1979 par le chercheur Nobuyuki Otsu. [5]

La méthode Otsu de seuillage reste toujours courante dans l'HTR [6] et elle fait partie de la librairie Python `numpy`. [7] Elle examine la variance entre les deux classes (*background* et *foreground*) pour déterminer un seuil idéal pour le jeu de données. Visualisées dans un histogramme, comme on voit dans la Figure 1.4, les données d'un jeu de pixels devraient se lever dans deux sommets et, idéalement, une baisse profonde devrait les diviser. Cette variance veut dire qu'il y a dans l'image une distinction importante entre les contours d'un caractère et l'arrière-plan de l'image. La méthode de seuillage examine la variance entre ces deux classes, le contour et l'arrière-plan, pour générer un seuil adapté aux données.

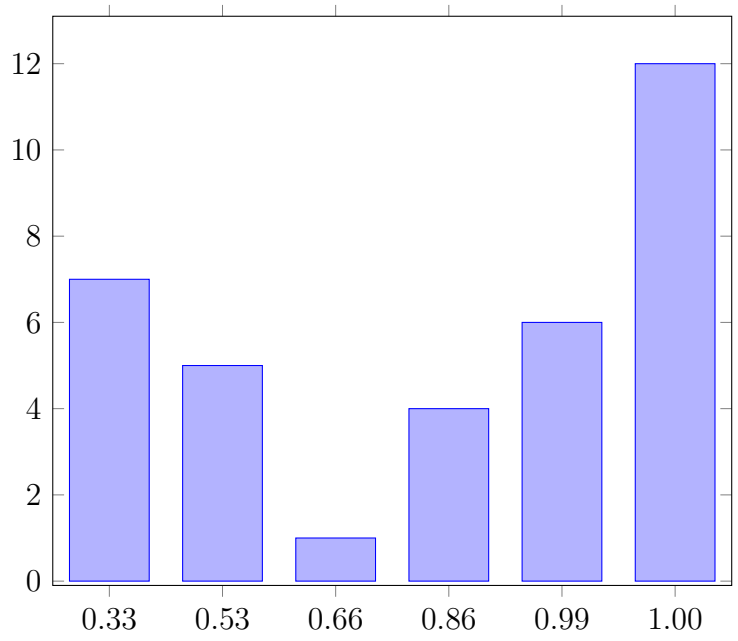


FIGURE 1.4 – Histogramme des valeurs niveau de gris des pixels

Le seuil sert à transformer la valeur numérique de chaque pixel soit en 0, soit en 1. Les pixels dont la valeur tombe au-dessous du seuil prennent la valeur 0 ; les autres, étant déterminés de faire partie du contour du caractère, prennent la valeur 1. La Figure 1.5c

5. Nobuyuki OTSU. « A Threshold Selection Method from Gray-Level Histograms ». In : *IEEE Transactions on Systems, Man, and Cybernetics* 9.1 (jan. 1979), p. 62-66.

6. Imran-Ahmed SIDDIQI, Florence CLOPPET et Nicole VINCENT. « Writing Property Descriptors : A Proposal for Typological Groupings ». In : *Gazette du livre médiéval* 56.1 (2011), p. 42-57. DOI : 10.3406/galim.2011.1981. URL : https://www.persee.fr/doc/galim_0753-5015_2011_num_56_1_1981 (visité le 02/08/2022).

7. *Numpy : NumPy Is the Fundamental Package for Array Computing with Python*. Version 1.23.1. URL : https://www.numpy.org (visité le 04/08/2022).

255	255	251.67	85	251.67
255	255	135	135	85
255	251.67	85	218.33	85
255	85	135	251.67	135
251.67	85	218.33	255	218.33
168.33	135	251.67	255	255
85	218.33	255	255	255

(a) La moyenne de chaque pixel

1.00	1.00	0.98	0.33	1.98
1.00	1.00	0.53	0.53	0.33
1.00	0.98	0.33	0.86	0.33
1.00	0.33	0.53	0.98	0.53
0.98	0.33	0.86	1.00	0.86
0.66	0.53	0.98	1.00	1.00
0.33	0.86	1.00	1.00	1.00

(b) L'image en niveau de gris

0	0	0	1	0
0	0	1	1	1
0	0	1	0	1
0	1	1	0	1
0	1	0	0	0
1	1	0	0	0
1	0	0	0	0

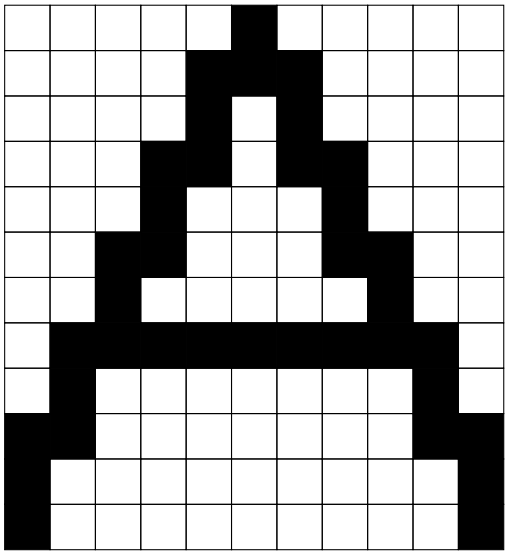
(c) L'image binarisée

FIGURE 1.5 – La binarisation

montre un exemple du résultat de ce triage. Ainsi, les logiciels OCR et HTR peuvent analyser les valeurs identiques et contiguës dans les données de l'image. Mais le logiciel n'est pas encore prêt à percevoir l'occurence d'un caractère.

Le masque

Ayant relevé sur l'image binarisée les points contiguës, c'est-à-dire les lignes, le logiciel doit ensuite reconnaître les objets qui s'en constituent. Les composants d'un caractère doivent s'encadrer dans un masque que le logiciel OCR ou HTR rendra comme une matrice binaire. (cf. Figure 1.6b) Grâce aux méthodes de segmentation, un logiciel produira les masques des glyphes, dont les composants qu'il a reconnus. En plus, le logiciel remarquera quels glyphes font partie de quels segments et quelles lignes de texte. Les processus de segmentation comptent sur la reconnaissance des espaces qui désunissent les groupes de points contiguës.



(a) Le masque de la lettre A,
sur l'image binarisée

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

(b) La matrice du masque de
la lettre A

1.2.2 L'intelligence artificielle

Pourquoi transforme-t-on les pixels d'un caractère en une matrice, telle que celle dans la Figure 1.6b ? À la base, l'ordinateur est une calculatrice. Les logiciels OCR et HTR décomposent une image numérique en les représentations mathématiques, les matrices. Ainsi, les matrices permettent aux logiciels faire des calculs probabilistes et prédire quel caractère correspond à quelle représentation. Mais pour faire des prédictions, un logiciel a besoin d'une intelligence artificielle.

Template matching

Pendant la dernière moitié du XXe siècle, la reconnaissance du texte et l'intelligence artificielle ont tous les deux vécu des progrès importants. Déjà en 1950, il y a plus que soixante-dix ans, Alan Turing a publié sa théorisation de l'IA. [8] Pendant les années soixante, au début de l'OCR, l'IA a rendu possible la prédiction du texte représenté par les matrices en correspondant la représentation qu'un logiciel a reconnue à la représentation d'un caractère qu'il avait dans sa base de données. Cette technique primitive est le *template matching* ou le filtrage par modèle. Elle n'est pas le moyen le plus pratique. Les chercheurs V. K. Govindan et A. P. Shivaprasad l'ont apprécié en 1990, après qu'elle a été dépassée par le *feature analysis*.

> [Template matching] directly compares an input character to a standard set of prototypes stored [*modèles stockés*]. The prototype that matches most closely provides recognition. […] This type of technique suffers from sensitivity to noise and is not adaptive to differences in writing style. [9]

Sous la dépendance du *template matching*, un logiciel OCR ne parviendra pas à la prédiction d'un caractère si la totalité de sa représentation en matrice n'est pas assez proche au modèle stocké lors de sa programmation.

Feature analysis

Les progrès dans l'IA pendant les années soixante-dix et quatre-vingt ont rendu possible le développement d'une nouvelle technique dont profitent toujours les logiciels OCR, quoique avec beaucoup d'élaboration depuis. Au lieu de correspondre la représentation d'un caractère à une autre, les logiciels OCR décomposent un caractère en ses *features* ou ses aspects. En 1990, Govindan et Shivaprasad ont résumé l'état de cette technique.

> The features may represent global and local properties of the characters. These include strokes, and bays in various directions, end points, intersections of line segments, loops […], and stroke relations, angular properties, sharp protrusions […]. These features have high tolerances to distortions and style variations, and also tolerate a certain degree of translation and rotation. However, the extraction processes are in general very complex and it is difficult to generate masks for these types of features. [10]

Ainsi, le logiciel OCR essaie de correspondre l'ensemble de certains aspects d'une représentation à l'ensemble des mêmes aspects aux lesquels le logiciel associe un caractère.

8. A. M. Turing. « Computing Machinery and Intelligence ». In : *Mind* LIX.236 (1er oct. 1950), p. 433-460. issn : 0026-4423. doi : 10.1093/mind/LIX.236.433. url : https://doi.org/10.1093/mind/LIX.236.433 (visité le 04/08/2022).

9. V. K. Govindan et A. P. Shivaprasad. « Character Recognition — A Review ». In : *Pattern Recognition* 23.7 (1990), p. 671. issn : 0031-3203. url : https://www.academia.edu/6986960/Character_recognition_A_review (visité le 05/08/2022).

10. Ibid.

Comme dissent Govindan et Shivaprasad, un aspect peut être un trait, le croisement des lignes, une boucle, la relation entre plusieurs traits, la caractéristique des angles, ou une saillie.

Grâce à la technique de *feature analysis*, un logiciel OCR moderne parvient à prendre une décision quant à la représentation d'un caractère bien qu'il n'ait pas trouvé le caractère de la même police, ayant la même représentation, dans sa base de données. *L'Handwritten Text Recognition* utilise aussi du *feature analysis*. Mais à la place de produire les métadonnées sur la police du caractère segmenté et reconnu ainsi que la langue du texte, l'HTR analyse les aspects d'un caractère simplement pour prédire le texte. Cependant, cette analyse n'est pas plus simple que celle réalisée par un logiciel OCR.

Le logiciel HTR a besoin d'associer beaucoup d'aspects d'une variété immense à un seul caractère puisqu'un main peut écrire un caractère par plusieurs moyens, selon la position de la lettre dans un mot. En outre, un main peut bien varier sa manière d'écriture et une page peut avoir plusieurs mains. Dit simplement, l'OCR analyse les aspects d'un caractère du point de vue d'une langue et d'une police attendue. L'HTR se focalise uniquement sur les aspects topologiques d'un caractère, sans besoin de savoir la langue ni avoir appris la police du texte.

Deep learning

Aujourd'hui l'HTR profite du *feature analysis* ainsi qu'un développement plus récent dans l'intelligence artificielle qui s'appelle le *deep learning* ou l'apprentissage profond. Pouvant s'améliorer grâce aux réseaux neurones, un logiciel OCR ou HTR moderne peut prendre les décisions de plus en plus bonnes quand il essaie de prédire du texte à partir de l'analyse des aspects (*feature analysis*). L'un des premiers mises en œuvre des reseaux neurones pour *l'Handwritten Text Recognition* était le projet european *Transkribus* en 2019.[11] Un autre projet européen a suivi *Transkribus* et, contraire au premier, laisse ses données et les architectures de ses modèles ouvertes : *Kraken*. Élaboré dans l'esprit de la science ouverte, le projet *Gallic(orpor)a* a profité plutôt du *Kraken* ainsi qu'une interface graphique qui le met en œuvre et qui est aussi ouverte, l'*eScriptorium*.

1.3 Le modèle HTR

Puisque *Transkribus* et *Kraken* profitent tous les deux de l'apprentissage profonde, les processus mis en œuvre par les interfaces graphiques *Transkribus* et *eScriptorium* ressemblent généralement à la même description. Ils commencent avec l'entraînement

11. Guenter Muehlberger et al. « Transforming Scholarship in the Archives through Handwritten Text Recognition : Transkribus as a Case Study ». In : *Journal of Documentation* 75.5 (9 sept. 2019), p. 954-976. ISSN : 0022-0418. DOI : 10.1108/JD-07-2018-0114. URL : https://www.emerald.com/insight/content/doi/10.1108/JD-07-2018-0114/full/html (visité le 04/08/2022).

d'un modèle HTR. Ensuite, ils appliquent le modèle entraîné aux données non préparées. Le taux de réussite se calcule par le pourcentage des mots et des lignes de texte que le modèle n'a jamais vus mais qui étaient bien prédits.

En premier temps, avant de penser à l'entraînement d'un modèle, il faut bien connaître sa saisie de données. Les données d'entré d'un modèle vont être les images numériques, composées des pixels. En général, leurs contenus textuels devraient se ressembler afin que le modèle se spécialise dans une écriture particulière. Il est pourtant possible de entraîner un modèle très généralisé. Cependant, il faut un corpus très large des données d'entraînement.

Le premier défi de l'entraînement d'un modèle HTR est la création des données d'entraînement. Lors de l'apprentissage, elles fournissent au modèle en cours son résultat souhaité pour qu'il puisse savoir comment se modifier sa manière de prédire afin de produire les bonnes prédictions. Les données doivent ressembler perfectement la prédiction idéale d'une donnée d'entrée. Grâce à l'apprentissage profonde, un modèle peut s'apprendre comment arriver à la prédiction souhaité, selon ses données d'entraînement.

Avec ces données d'entraînement, qui se connaissent parfois comme le corpus d'or puisqu'elles sont générées avec beaucoup de soin et souvent beaucoup d'argent, un modèle a aussi besoin des données non préparées qui ressemblent aux données préparées du corpus d'or. Le modèle HTR va s'évalue lors de l'apprentissage selon son taux de réussite avec les données qu'il devrait savoir prédire, les données préparées, et les données qu'il pense de n'avoir jamais vu, les données non préparées qui font partie de ce qu'on appelle un *training set*. Il est le combinaison de ces deux jeux de données, préparées et non préparées, qui rend possible l'apprentissage profonde.

Le résultat de l'entraînement s'appelle un modèle. Étant spécialisé dans une écriture particulière, le modèle pourrait ensuite se mettre en œuvre pour prédire du texte sur les images qui ressemblent aux données d'entraînement. Les chercheurs s'occupent du développment des nouveaux modèles HTR qui se spécialisent dans certains manuscrits et certaines imprimés historiques. Le projet *Gallic(orpor)a* visait à entraîner plusieurs modèles. Un se spécialiserait dans l'écriture des manuscrits du XVe siècle, l'un dans l'écriture des incunables du XVe siècle, un autre dans l'écriture des imprimés du XVIe siècle, un autre dans celui des imprimés du XVIIe siècle, et un autre dans l'écriture des imprimés du XVIIIe siècle.

Cependant, tous les modèles du projet *Gallic(orpor)a* se spécialisent dans les écritures latines. En théorie, un modèle HTR n'est pas limité aux polices et aux langues. Mais, jusqu'au présent, tout modèle efficace compte sur une écriture specifique. Donc, un modèle entraîné sur les imprimés XVIIe siècle en français peut cependant produire des bonnent prédictions du texte d'une imprimée du même siècle en italien. Par contre, un texte écrit en arabe ne parviendra pas au même taux de réussite. Le modèle ne sait pas segmenter la ligne de texte ni décomposer ses segments en glyphes puisqu'il s'est appris

comment reconnaître des caractères en cherchant des espaces attendues dans une écriture latine.

Chapitre 2

Au commencement, il y avait les *guidelines SegmOnto*

2.1 La problématique

Un manuscrit se définit par ses moyens de création. Étant rédigé à la main, souvent avant la croissance de l'imprimerie, un manuscrit est un objet singulier dans le monde. Il est vrai que d'autres ressources peuvent présenter le même texte, les mêmes images, ou bien la même musique que présent un manuscrit. Cependant, un manuscrit n'a pas d'autre exemplaire. Ses contenus sont réalisés par et se répandent dans sa constitution matérielle particulière. Un manuscrit est unique avec son écriture, parfois de plusieurs mains, ses fautes d'écriture, ses parties abîmées, décorées, ou révisées, sa provenance et son histoire en tant qu'objet rare qui se transfère entre des individus, des familles, et des organisations. De plus, chaque propriétaire peut encore modifier la ressource en ajoutant ou retirant des pages, en corrigeant ou blâmant du texte ou des images, ainsi qu'en changeant la reliure et les informations portant sur l'édition.

Pour étudier un manuscrit, il faut donc développer un vocabulaire qui sait décrire les divers aspects de l'objet composé. Après tout, le pouvoir de bien définir les termes d'une étude est à la base de l'analyse. Sans un vocabulaire bien élaboré et cohérent, les arguments d'une chercheuse ou d'un chercheur ne seront pas compréhensibles. L'harmonisation d'un vocabulaire est encore plus importante eu égard à la communication des découverts et à la collaboration entre plusieurs personnes, surtout si elles ont des spécialités différentes. Ces deux activités, la communication et la collaboration, sont fondamentales à la recherche et exigent donc l'élaboration d'un vocabulaire cohérent pour décrire des manuscrits.

Voici les défis lexicographiques de l'étude des manuscrits et voici l'un des obstacles que le projet *Gallic(orpor)a* a essayé de franchir. Imaginons, par exemple, qu'on veut décrire le texte principal qui se trouve sur la page d'un document. On peut y appliquer l'étiquette descriptive *Main Zone* ou bien *Principal Text*. En lisant des articles scienti-

fiques où chacun utilise une étiquette différente, un humain arriverait à reconnaître que les étiquettes différentes parlent de la même partie de la page. Mais pour un outil informatique, si la même région d'une page ne porte pas la même étiquette, il n'arriverait pas à les associer sans être instruit à chercher plusieurs versions du même concept. Dans ce cas, l'analyse serait trop compliquée à effectuer à l'échelle. C'est ainsi que la description des manuscrits est importante au projet *Gallic(orpor)a*. Le projet *Gallic(orpor)a* cherchait donc à profiter d'un vocabulaire bien élaboré et cohérent, qui pourrait décrire soit les manuscrits, soit les imprimés historiques. Il y avait plusieurs projets qui avaient cherché à répondre à cette problématique, mais celui que le projet *Gallic(orpor)a* a choisi est le vocabulaire du projet *SegmOnto*.

2.2 Les solutions proposées

Plusieurs projets avaient proposé des solutions quant à la description normalisée des manuscrits. Hors de la France, les vocabulaires ont été élaborés notamment en anglais et pour les manuscrits médiévaux. Un exemple important est la base de données *DigiPal* (Digital Resource and Database of Palaeography, Manuscripts and Diplomatic), qui n'est plus mis à jour mais qui a été développé au sein du département des humanités numériques à King's College London. [1] L'un de ses auteurs, Peter Stokes, travaille actuellement en France et continue dans la même veine en contribuant au projet *SegmOnto* qui était développé dans un environnement francophone, même si son vocabulaire est rédigé en anglais. [2] Mais le projet *SegmOnto* n'est pas le premier projet français qui a essayé d'élaborer un lexique pour les documents historiques. Ni est le projet *DigiPal* le premier en Europe. Avant leur création, la codicologie en France et en Europe suivait le modèle de Denis Muzerelle et son *Vocabulaire codicologique*, que nous expliquons prochainement.

2.2.1 Le Vocabulaire international de la codicologie

En 1985, Denis Muzerelle a conçu un vocabulaire codicologique qui avait pour but de fournir des médiévistes avec des termes uniformes pouvant décrire les aspects d'un manuscrit. [3] Depuis l'apparition de son vocabulaire en français, des autres chercheurs sont venus pour adapter les termes de Muzerelle en d'autres langues. Marilena Maniaci

1. *DigiPal : Digital Resource and Database of Palaeography, Manuscripts and Diplomatic.* London, été 2011. URL : `http://www.digipal.eu/`.

2. Simon GABAY, Jean-Baptiste CAMPS et Ariane PINCHE. « SegmOnto ». In : *Création de Modèle(s) HTR Pour Les Documents Médiévaux En Ancien Français et Moyen Français Entre Le Xe-XIVe Siècle*. Paris, France : Ecole nationale des chartes | PSL, nov. 2021. URL : `https://hal.archives-ouvertes.fr/hal-03481089` (visité le 25/07/2022).

3. Dennis MUZERELLE. *Vocabulaire Codicologique : Répertoire Méthodique Des Termes Français Relatifs Aux Manuscrits*. Rubricae 1. Paris : Éd. Cemi, 1985.

a publié une version du *Vocabulaire codicologique* pour l'italien en 1996. [4] Pilar Ostos, Luisa Pardo, et Elena Rodríguez en ont créé un pour l'espagnol l'année suivante. [5] Parfois appelé le *Vocabulaire international de la codicologie*, l'édition multilingue du *Vocabulaire codicologique* que Muzerelle a commencé en 1985 était maintenue jusqu'à l'édition d'une version 1.1. en 2002-2003. (besoin de citation)

2.2.2 La Codicologia

Aujourd'hui, la paléographie et l'étude des manuscrits peuvent profiter de l'application web *Codicologia* qui réunit le *Vocabulaire codicologique* ainsi que deux autres bases de données similaires : le projet multilingue *Lexicon* et le *Glossaire codicologique arabe*. Ses trois bases de données spécialisent dans divers écritures. Le *Vocabulaire codicologique* a été développé pour les manuscrits de l'écriture latin. Piloté par Philippe Bobichon, le projet *Lexicon* présente un vocabulaire en français pour décrire les manuscrits écrits en latin, roman, grec, hébreu, et arabe. [6] Un vocabulaire spécialisé plus profondément pour l'arabe a été élaboré dans le *Glossaire codicologique arabe* d'Anne-Marie Eddé et Marc Geoffroy. [7] Ce dernier a été conçu au sein de l'Institut de recherche et d'histoire des textes après les modèles de Muzerelle et le vocabulaire codicologique en arabe d'Adam Gacek. [8]

L'application web *Codicologia* rassemblent ces projets et présente un vocabulaire bien étendu. Par exemple, *Codicologia* fournit 15 termes pour décrire une faute d'écriture dans un manuscrit. Certains de ses termes possèdent eux-même plusieurs définitions que les divers bases de données fournissent. Le terme *caviarder*, par exemple, a une définition courte dans le vocabulaire français de Muzerelle.

> Supprimer un mot, un passage..., en le recouvrant largement d'encre, de façon à ce qu'il ne puisse être lu. [9]

Selon le *Lexicon* de Bibichon, par contre, le *caviarder* se définit d'une manière plus détaillé et vise à expliquer l'étymologie du mot afin de préciser son usage dans le cadre des manuscrits des divers écritures.

---

4. Marilena Maniaci. *Terminologia Des Libro Manoscritto*. Addenda 3. Rome : Istituto centrale per la patologia del libro, 1996.

5. Pilar Ostos, M. Luisa Pardo et Elena E. Rodríguez. *Vocabulario de codicologÍa : Versión Española Revisada y Aumentada Del Vocabulaire Codicologique*. Instrumenta Bibliologica. Madrid : Arco/Libros, 1997.

6. Philippe Bobichon. « Le Lexicon : Mise En Page et Mise En Texte Des Manuscrits Hébreux, Grecs, Latins, Romans et Arabes ». In : (2009), p. 81. URL : `https://cel.archives-ouvertes.fr/cel-00377671` (visité le 25/07/2022).

7. « Glossaire codicologique français-arabe ». In : *Gazette du livre médiéval* 40.1 (2002), p. 79-80. URL : `https://www.persee.fr/doc/galim_0753-5015_2002_num_40_1_1563` (visité le 25/07/2022).

8. Adam Gacek. *The Arabic Manuscript Tradition : A Glossary of Technical Terms and Bibliography*. Handbook of Oriental Studies 1, The Near and Middle East. Leiden : Brill, 2001. 269 p. ISBN : ISBN 90-04-12061-0.

9. Denis Muzerelle. *Caviarder*. In : *Codicologia*. Institut de recherche et d'histoire des textes, 2011. URL : `http://codicologia.irht.cnrs.fr/theme/liste_theme/413#tr-868` (visité le 25/07/2022).

> Le mot [*caviarder*] apparaît en 1907 (noircir à l'encre) : il désigne alors un procédé appliqué par la censure russe, sous Nicolas Ier. Dans certains manuscrits grecs, le détail rempli d'encre est surmonté d'un point et d'un trait court destinés à le neutraliser. Ce procédé est très souvent utilisé parmi d'autres, pour la censure des manuscrits hébreux effectué sous l'autorité de l'inquisition, en Italie, à la fin du xvie siècle et au début du xviie. [10]

Étant élaboré à partir d'un corpus très diversifié, le *Lexicon* de Bibichon a moins de termes qu'a le *Vocabulaire codicologique* mais ses termes sont plus généralisés. Le vocabulaire de Muzerelle, par contre, fait plus de distinctions entre les aspects d'un manuscrit et donc a plus de termes distincts par rapport aux deux autres vocabulaires de l'application *Codicologia*.

En réunissant les trois bases de données, sans privilégier aucun, *Codicologia* présente un vocabulaire codicologique vraiment vaste. Cependant, l'application *Codicologia*, comme toutes ses bases de données, vise à répondre au manque de cohérence dans la manière par laquelle la communauté scientifique décrit les manuscrits. Le grandeur de son vocabulaire pose un problème à cet objectif. Ayant plus de deux milles termes en français—certains d'entre eux ont eux-même plusieurs définitions—la solution proposée par *Codicologia* livre un vocabulaire bien harmonisé et documenté mais trop étendu pour être appliqué à l'échelle dans une approche informatique.

Sans un corpus d'entraînement gigantesque, qui couterait une somme énorme, l'apprentissage automatique ne peut pas faire de distinction au niveau des termes conçus par Muzerelle et les autres auteurs des bases de données de *Codicologia*. Aujourd'hui, un modèle ne peut pas s'entraîner sur des milles des étiquettes possibles et arriver à distinguer entre, par exemple, 15 types de faute d'écriture. Un humaine peut le faire, et pour cette raison les bases de données de *Codicologia* sont utiles. Mais leurs vocabulaires ne conviennent pas bien à une approche informatique.

2.3 Les *guidelines* de *SegmOnto*

Le projet *SegmOnto* propose un vocabulaire plus petit qui peut pourtant décrire une grande diversité de documents historiques, y compris les manuscrits et les imprimés. Cet objectif est encore plus compliqué à achever qu'un vocabulaire spécialisé aux manuscrits. Décrire les documents d'une diachronie longue, et sans préférence d'une écriture en particulier, exige un équilibre délicat entre la généralité et la particularité. Pour y arriver, les *guidelines* du projet *SegmOnto* limite le nombre de termes dans son vocabulaire, sans en priver aucun d'une identité distincte.

10. Philippe Bobichon. *Caviarder*. In : *Codicologia*. Institut de recherche et d'histoire des textes, 2011. URL : `http://codicologia.irht.cnrs.fr/theme/liste_theme/413#tr-868` (visité le 25/07/2022).

Les *guidelines* se divisent en deux catégories : les « zones » et les « lines ». La première parle des régions sur la page, y compris les régions de texte et les régions sans texte, tel qu'une image. Pour la plupart de temps, la catégorie de la ligne veut décrire les différents types de lignes de texte. Mais une ligne du vocabulaire *SegmOnto* peut aussi tracer la ligne d'une partition musicale ou une ligne réelle sur la page qui n'oriente pas des autres systèmes d'écriture, telle qu'une ligne qui divise la page en deux. Chacune de ces deux catégories se compose d'une liste des étiquettes, et chacune d'elles cherche à parvenir à l'équilibre entre la généralité et la particularité. Une étiquette devrait pouvoir être appliquée à soit un manuscrit, soit un imprimé, de peu importe quelle langue et quelle écriture.

2.3.1 Les zones

— **CustomZone** : une zone qui ne convient pas à aucune d'autres catégories de zone.
— **DamageZone** : une zone qui contient des marques de dégâts sur le document source, tel qu'un trou.
— **DecorationZone** : une zone qui contient un élément graphique, y compris de la peinture et les petits dessins dans la marge de la page.
— **DigitizationArtefactZone** : une zone qui contient un item qui n'appartient pas au document source mais est lié au processus de la numérisation, tel qu'une règle pour montrer la mesure du document.
— **DropCapitalZone** : une zone qui contient une initiale ; l'initiale peut prendre l'espace de plusieurs lignes de texte ou porter une décoration importante, tel que de l'historicisation, l'ornementation, ou des dessins.
— **MainZone** : une zone qui contient le texte principal du document source.
— **MarginTextZone** : une zone qui contient le texte dans la marge du document source.
— **MusicZone** : une zone qui contient une partition musicale.
— **NumberingZone** : une zone qui contient des numéros de page, y compris les numéros rédigés en chiffres romans.
— **QuireMarksZone** : une zone qui contient des notes en bas page destinées à la fabrication du document source pour garder les pages dans le bon ordre.
— **RunningTitleZone** : une zone qui contient une version du titre du document ou d'une section du document qui se trouve en tête de la page.
— **SealZone** : une zone qui contient un sceau sur le document source.
— **StampZone** : une zone qui contient l'empreint d'un tampon sur le document source.
— **TableZone** : une zone qui contient une table.

— **TitlePageZone** : une zone souvent sur l'une des premières pages du document source qui contient toutes les informations concernant le titre et l'édition du document.

2.3.2 Les lignes

— **CustomLine** : une ligne qui ne convient pas à aucune d'autres catégories de ligne.
— **DefaultLine** : une ligne qui contient du texte attendu dans la zone.
— **DropCapitalLine** : une ligne qui contient l'initiale.
— **HeadingLine** : une ligne qui contient le texte d'un titre, tel que celui d'une section ou d'un chapitre.
— **InterlinearLine** : une ligne qui traverse la page pour marquer une limite.
— **MusicLine** : une ligne de la portée d'une partition.

Chapitre 3

Le rêve du projet *Gallic(orpor)a*

3.1 Le contexte du projet

Présenter les institutions qui soutiennent le projet (École nationale des chartes, Inria, Université de Genève, DataLab de la BnF) et les projets qui l'ont précédé et sur lesquels *Gallic(orpor)a* compte.

3.2 L'objectif du projet

Le projet a pour but d'arriver des images numérisées à un document numérique sans besoin d'un éditeur. L'idée est qu'avec le produit du pipeline, qui sera un texte prédit et pré-éditorialisé, un individu qui n'est pas forcement spécialisé dans l'informatique peut facilement le prendre et effectuer des analyses et/ou éditer le texte du document dont les images de ses pages ont été traités.

3.3 Le pipeline

Présenter le pipeline. Réutiliser l'information qu'on a générée pour le présenter à la BnF.

Deuxième partie

Exposition de la préparation et du travail d'analyse

Chapitre 4

Un pipeline visant à tout rassembler

4.1 La reconnaissance du texte et des segments

Le premier étape du pipeline *Gallic(orpor)a* est la reconnaissance du texte sur les images numérisées. Pour arriver d'un rassemblement de pixels au caractère d'un système d'écriture, il faut un modèle HTR qui sait chercher dans les pixels les configurations des caractères. Avec la reconnaissance de texte, il faut un deuxième modèle qui sait reconnaître les régions cohérentes sur la page. Ce dernier modèle cherchent aussi dans les pixels pour les configurations consistantes, mais au lieu de reconnaître dedans des caractères, il relève les polygons ou les rectangles qui contient une entité cohérente.

4.1.1 La création des données d'entraînement

4.1.2 L'entraînement des modèles

4.1.3 Les modèles prédisent le texte

4.2 La reconstitution des données

4.3 L'analyse linguistique

4.3.1 La création des données d'entraînement

4.3.2 L'entraînement des modèles

4.3.3 Les modèles analysent le texte prédit

4.4 Le texte pré-édité

Chapitre 5

L'analyse des structures des données XML

5.1 XML-ALTO

5.1.1 Qu'est-ce qu'est le format ALTO ?

Décrire la création, le suivi, et l'objectif du format XML ALTO : enregistrer les infos sur la structure d'une image segmentée.

5.1.2 La structure des fichiers XML-ALTO

Montrer la structure des données d'un fichier ALTO dans les deux formats qui sortent de Kraken : (1) ligne de texte encodé dans la balise `<TextLine>`, qui sort de Kraken via l'interface d'eScriptorium, et (2) ligne de texte encodé au niveau du glyph, qui sort directement de la ligne de commande de Kraken.

5.2 XMl-TEI

5.2.1 Qu'est-ce qu'est la TEI ?

Décrire la création, le suivi, et l'objectif de la TEI.

5.2.2 Les éléments de base de la TEI

Expliquer qu'il y a deux éléments essentiels de la racine, le `<teiHeader>` et le `<body>`. Ensuite expliquer l'utilité de l'élément facultatif `<sourceDoc>` et expliquer pourquoi il convient bien aux données de structure d'un fichier ALTO.

Chapitre 6

À la recherche des métadonnées

6.1 Uniquement l'essentiel

6.1.1 Documents de divers types et de plusieurs époques

Expliquer le défi de modéliser un `<teiHeader>` qui est à la fois assez généralisé pour convenir aux divers documents et assez précisé pour servir aux utilisateurs et à la recherche.

6.1.2 Exemples des métadonnées souhaitées

Donner un exemple des métadonnées d'un imprimé (cf. fig. 6.1) :

```
<sourceDesc>
  <biblStruct>
    <monogr>
      <author xml:id="author">
        <persName>
          <surname>Balzac</surname> <!-- auteur -->
          <forename>Honoré</forename>
        </persName>
      </author>
      <title>The Wild Ass's Skin</title> <!-- titre -->
      <editor role="translator">Ellen Marriage</editor>
      <editor role="preface">George Saintsbury</editor>
      <pubPlace key="FR">Paris</pubPlace>
      <imprint>
        <pubPlace>London</pubPlace> <!-- lieu de publication -->
        <publisher>Dent</publisher> <!-- éditeur -->
        <date when="1906">1906</date> <!-- date de publication -->
      </imprint>
    </mongr>
  </biblStruct>
```

FIGURE 6.1 – Exemple des métadonnées d'un imprimé encodées en TEI (emprunté de `teibyexample.org` – à changer)

Et donner un exemple d'un incunable (cf. fig. 6.2) :

```
<sourceDesc>
  <msDesc>
    <msContents>
      <biblStruct>
        <monogr>
          <author xml:id="author">
            <persName>
              <surname>Tory</surname> <!-- auteur -->
              <forename>Geoffroy</forename>
            <persName>
          </author>
          <title>Champ fleury</title> <!-- titre -->
          <imprint>
            <pubPlace>Paris</pubPlace> <!-- lieu de publication / lieu d'
    apparition -->
            <publisher>
              <persName>
                <surname>Gourmont</surname> <!-- éditeur -->
                <forename>Gilles de</forename>
              </persName>
            </publisher>
          </imprint>
        </mongr>
      </biblStruct>
```

FIGURE 6.2 – Exemple des métadonnées d'un incunable encodées en TEI (emprunté du cours TEI de J-B Camps 2015 – à changer)

Expliquer comment l'objectif du projet *Gallic(orpor)a* de traiter des documents d'une diachronie longue pose un défi à la récupération et encodage généralisée des métadonnées.

6.2 Où se trouvent les métadonnées des sources de Gallica

Présenter les deux sources de métadonnées ciblées par l'application `alto2tei`.

6.2.1 L'IIIF Image API

L'API du manifest IIIF contient des données rudimentaires sur le document. Elles sont envoyées dans un format JSON. Donner un exemple (cf. fig. 6.3) :

6.2.2 L'API SRU de la BnF

L'API du catalogue général de la BnF contient des données bien précises sur le document. Elles sont envoyées dans un format XML-Unimarc. Donner un exemple (cf. fig. 6.4).

```
{"Metadata":
    {
    "Label": "Title",
    "Value": "The Wild Ass's Skin", # titre

    "Label": "Creator",
    "Value": "Honoré Balzac" # auteur
    }
}
```

FIGURE 6.3 – Exemple des métadonnées envoyées par l'API IIIF

```
  <mxc:datafield tag="200" ind1="1" ind2=" ">
    <mxc:subfield code="a">The Wild Ass's Skin</mxc:subfield> <!-- titre --
    >
    <mxc:subfield code="b">Texte imprimé</mxc:subfield>
  </mxc:datafield>
  <mxc:datafield tag="210" ind1=" " ind2=" ">
    <mxc:subfield code="a">Londres</mxc:subfield> <!-- lieu de publication
    -->
    <mxc:subfield code="c">Dent</mxc:subfield> <!-- éditeur -->
    <mxc:subfield code="d">1906</mxc:subfield> <!-- date de publication -->
  </mxc:datafield>
[...]
  <mxc:datafield tag="700" ind1=" " ind2="|">
    <mxc:subfield code="a">Balzac</mxc:subfield> <!-- auteur -->
    <mxc:subfield code="b">Honoré</mxc:subfield>
  </mxc:datafield>
```

FIGURE 6.4 – Exemple des métadonnées envoyées par l'API SRU de la BnF

6.3 Une solution

Présenter les métadonnées du document qu'on a déterminé d'être essentielle / assez généralisées parmi les divers types de document.

Troisième partie

Mise en opérationnelle du projet

Chapitre 7

La génération du `<teiHeader>`

7.1 La récupération des données

Parler de la récupération des données depuis les deux APIs discutés (cf. fig. **??**) ainsi que le fichier YAML de configuration.

7.1.1 Du manifest IIIF à un dictionnaire Python

Montrer le mapping des données du manifest au dictionnaire Python.

7.1.2 De l'Unimarc à un dictionnaire Python

Montrer le mapping des données du catalogue au dictionnaire Python.

7.2 L'analyse des données

Parler de la stratégie d'atténuation des risques en sélectionnant les données fiables. Les métadonnées du catalogue général de la BnF sont utilisées uniquement si le même exemplaire physique du document numérisé sur Gallica a bien été trouvé. Sinon, on risque de mettre les données d'un autre exemplaire de l'oeuvre que celui qui a été transcrit et donc introduire des fausses données, tel que le cote ou même l'éditeur et la date de publication de l'exemplaire.

7.3 Le modèle du `<teiHeader>`

Montrer le mapping des données au `<teiHeader>`.

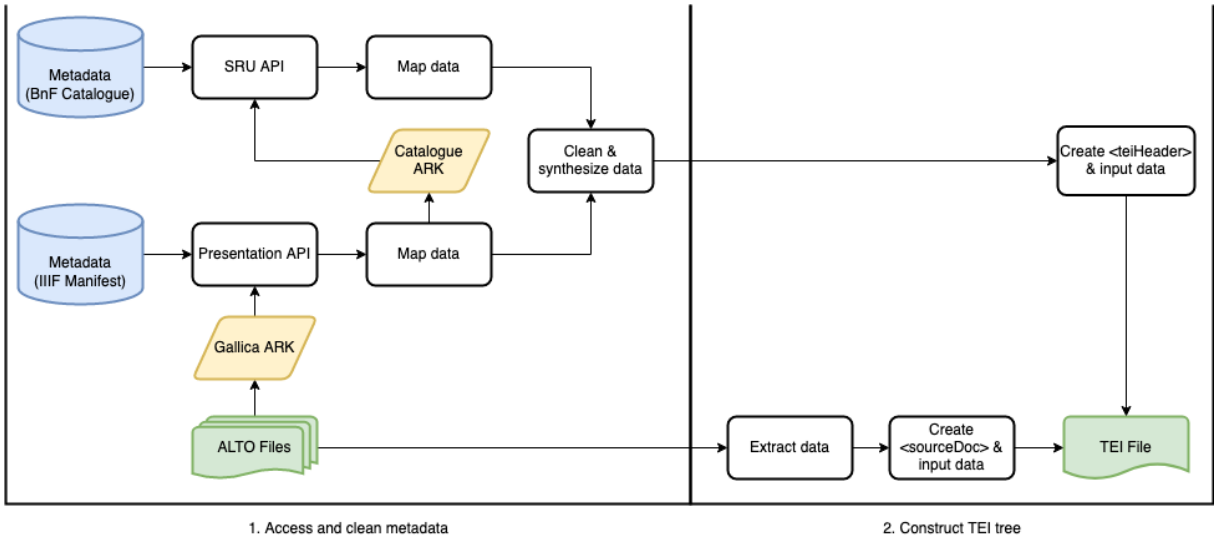


FIGURE 7.1 – Workflow

Chapitre 8

La modélisation de la <sourceDoc>

8.1 Le modèle du <sourceDoc>

Montrer le mapping des données prédites par les modèles HTR et segmentation vers les éléments TEI de la <sourceDoc>.

8.1.1 Niveau de la ligne de texte

Des exemples / tables ...

8.1.2 Niveau d'un mot ou d'une espace

Des exemples / tables ...

8.1.3 Niveau du glyphe

Des exemples / tables ...

8.2 Documenter ce modèle dans l'ODD

Présenter le travail d'avoir écrit l'ODD *SegmOnto-Gallicorpora*.

Chapitre 9

Les données textuelles produites

9.1 La génération du `<body>` grâce au lexique *SegmOnto*

Expliquer comment j'ai généré le `<body>` en prenant les lignes de texte de la `<sourceDoc>` selon leurs étiquettes.

9.2 L'analyse linguistique

Parler d'un travail futur qui pourra prendre le fichier XML-TEI que j'ai créé et extraire les lignes de texte du `<body>` et les passer aux modèles TAL pour faire des analyses linguistiques. Montrer un exemple de ces données et comment cela marcherait avec un travail préliminaire que j'aurai fait d'ici la fin de stage et/ou le travail d'un ancien stagiaire que j'ai appliquée aux données Gallic(orpor)a.

Conclusion

Annexe A

Données

Données du projet.

Glossaire

Handwritten Text Recognition La reconnaissance du texte écrit sur une image numérique. 3

Optical Character Recognition La reconnaissance des polices du texte sur une image numérique. 5

Acronymes

HTR Handwritten Text Recognition. 3–10, 12, 13

OCR Optical Character Recognition. 5, 6, 9–12

Table des figures

Liste des tableaux

Table des matières