h e g

Haute école de gestion
Genève

Ex imagine ad litteras :

Projet d'océrisation de la collection De Bry

Bodmer Lab
Digital World Literature

Mémoire de recherche réalisé dans le cadre du

Master en Sciences de l'Information

par :

Florence BURGY, Steeve GERSON et Loïc SCHÜPBACH

Sous la direction de **Dr Julien GOBEILL**

Genève, le 16 janvier 2020

Haute École de Gestion de Genève (HEG-GE)

Filière ID

Hes·so GENÈVE
Haute Ecole Spécialisée
de Suisse occidentale

Résumé

Ce projet de recherche, en collaboration avec le Bodmer Lab, consiste à océriser la collection de Bry, des imprimés latins des XVIe et XVIIe siècles, afin d'en obtenir une transcription aussi correcte que possible et de la rendre explorable par la recherche plein texte.

Dans un premier temps, nous avons testé quatre logiciels d'océrisation gratuits et open source, Tesseract, Kraken, Calamari et OCR4all. Kraken et Calamari ont donné des résultats peu convaincants, mais Tesseract et OCR4all étaient bien plus performants. Nous avons testé en mesurant la précision, le rappel, et la F-mesure (F1) au niveau des caractères et au niveau des mots. Pour Tesseract, nous avons obtenu une F1 de 78.62% (caractères) et 31.78% (mots). Pour OCR4all, nous avons obtenu une F1 de 85.43% (caractères) et 49.51% (mots). Cependant, un bug d'OCR4all en rend son utilisation complexe et chronophage, et nous avons choisi de travailler avec Tesseract.

Nous avons ensuite essayé différentes méthodes pour améliorer les résultats obtenus avec Tesseract, certaines basées sur le traitement des inputs, d'autres sur le traitement des outputs, et une autre sur une fonctionnalité du logiciel. Toutes les méthodes n'étaient pas nécessairement efficaces, mais grâce à certaines, nous avons pu atteindre une F1 de 80.06% au niveau des caractères et de 34.58% au niveau des mots.

Enfin, nous avons proposé des solutions d'intégration des transcriptions sur le site web du Bodmer Lab. Nos recommandations prennent en compte les technologies actuellement utilisées par le mandant, à savoir IIIF et Mirador, et se basent sur des méthodes en usage dans des institutions similaires.

Bodmer Lab – OCR – Reconnaissance Optique de Caractères – Intelligence Artificielle – Tesseract – De Bry – Latin – Humanités numériques

Table des matières

Liste des figures

1. Le projet en bref

1.1 Contexte

Dans le cadre du Master en Sciences de l'Information de la Haute École de Gestion de Genève, nous avons eu l'occasion de mener un projet de recherche sur deux semestres. Notre groupe a choisi de travailler sur un sujet proposé par le Bodmer Lab.

Le Bodmer Lab se définit comme "un projet de recherche et de numérisation issu d'un partenariat entre l'Université de Genève et la Fondation Martin Bodmer" (Bodmer Lab 2019), dont l'objectif est de rendre accessible à un large public des documents provenant de la *Bibliotheca Bodmeriana*. Cette collection, que Martin Bodmer voulait "bibliothèque de la littérature mondiale" (Bodmer Lab 2019) au sens où l'entendait Goethe, regroupe en effet de nombreux ouvrages anciens, rares et fragiles, dont la valeur historique est difficilement égalable. La numérisation de ces documents permet de les faire connaître et de les rendre exploitables par des chercheurs comme par le grand public. Un important travail de mise en valeur et de médiation complète ce processus.

1.2 Texte

Au sein de la *Bibliotheca Bodmeriana*, certains ensembles de documents présentent une structure suffisamment cohérente pour pouvoir être considérés comme des sortes de sous-collections. Ce fait a mené le Bodmer Lab à traiter ces ensembles, ou "constellations", de manière indépendante et à en confier l'étude à des spécialistes des domaines concernés (Bodmer Lab 2019).

L'une de ces "constellations", la collection De Bry, qui rassemble des récits de voyage de l'époque des Grandes découvertes, est l'objet de ce projet de recherche. Cette collection exceptionnelle de vingt-neuf volumes illustrés par des gravures et datant des XVIe et XVIIe siècles est divisée en deux parties : les « Grands Voyages », ou *India occidentalis*, qui retracent l'exploration des Amériques, et les « Petits Voyages », ou *India orientalis*, qui concernent essentiellement les voyages en Afrique et en Asie. Ces ouvrages sont l'œuvre de l'éditeur-imprimeur liégeois Théodore de Bry (1528-1598) et de ses descendants. Ces volumes rares sortent de leur atelier de Francfort entre 1590 et 1634 et sont édités en plusieurs langues (Bodmer Lab 2019).

La première édition latine de cette collection, que la Fondation Bodmer possède dans son intégralité, est déjà numérisée et accessible en ligne sur le site du Bodmer Lab (Bodmer Lab 2019), mais il est impossible pour le moment d'y effectuer une recherche plein texte.

1.3 Objectifs

Cette problématique est au cœur de ce projet de recherche, dont le but premier est en effet d'océriser la collection De Bry. L'océrisation (de l'acronyme OCR, pour *Optical Character Recognition*, reconnaissance optique de caractères en français) est une technologie qui permet d'identifier des caractères dans un document numérisé au format image et de les extraire dans un format texte lisible par un humain comme par une machine, de manière totalement automatisée.

Dans ce projet, notre travail a consisté essentiellement à utiliser un logiciel d'océrisation, à l'entraîner et à en optimiser les paramètres afin d'obtenir une transcription automatique des textes numérisés qui soit au plus proche de l'original.

Plus précisément, nous avons dans un premier temps testé et comparé quatre logiciels d'océrisation afin de sélectionner celui qui s'adaptait le mieux à nos besoins. Pour des raisons financières et de faisabilité, nous nous sommes tournés vers des logiciels libres et *open source*, afin de pouvoir aisément avoir accès au code et en modifier les paramètres.

Une fois le logiciel sélectionné, nous avons fait en sorte d'en optimiser les paramètres et de l'entraîner avec des jeux de données adéquats afin d'obtenir les transcriptions les plus correctes possible. Nous avons donc dû évaluer la qualité du texte extrait à chaque itération, dans le but d'obtenir la transcription la plus fidèle possible. Nous espérions obtenir un taux de réussite de 95% pour la transcription complète mais, au terme de ce projet, nous n'avons pu obtenir qu'un peu plus de 80% de caractères corrects et environ 34% de mots corrects, ce qui reste honorable.

En outre, selon les souhaits du mandant, nous avons réfléchi à des solutions pour intégrer ces transcriptions sur l'interface de la constellation et les rendre exploitables par toute personne désirant explorer cette collection par la recherche plein texte. Nous proposons donc trois solutions différentes, basées sur les outils déjà mis en place par le Bodmer Lab, à savoir les spécifications techniques IIIF et des plateformes de visualisation qui leur sont liées.

2. État de l'art

Avant de se lancer dans la recherche à proprement parler, il est essentiel d'effectuer un état de l'art de la discipline concernée afin de pouvoir situer notre projet dans son contexte scientifique. Celui-ci s'articulera selon deux axes. Le premier présentera la technologie OCR dans son état actuel en en retraçant brièvement l'histoire, en en expliquant le fonctionnement et en présentant des méthodes d'évaluation des logiciels OCR. Le second axe présentera les développements récents dans le domaine de l'océrisation, et se focalisera plus précisément sur la problématique des documents anciens et des langues anciennes, qui concerne directement notre projet de recherche.

2.1 OCR – une technologie mature

2.1.1 Origines et développement de l'OCR

La technologie connue sous le nom d'OCR a vu le jour dans la première moitié du XXème siècle. Malgré de premiers développements intéressants, l'idée de créer des machines capables de lire les caractères et les chiffres est restée un rêve jusque dans les années 1950 (Mori, Suen, Yamamoto 1992). C'est à cette époque que l'OCR trouve son marché et se développe non seulement comme technologie mais comme produit commercial, sous l'impulsion, entre autres, de David Shepard, fondateur de *Intelligent Machines Research Corporation* (Nagy 2016).

Les progrès dans ce domaine sont rapides, et il serait surfait de citer toutes les recherches entreprises au cours des soixante dernières années. Mentionnons cependant la machine de Jacob Rabinow, développée dans les années 1960 et permettant de lire et trier les adresses postales américaines, et celle de Kurzweil, dans les années 1970, permettant la reconnaissance et la lecture de textes aux aveugles (Nagy 2016). Un historique plus complet des développements dans ce domaine à cette époque se trouve dans l'ouvrage de Herbert F. Schantz, *The history of OCR, optical character recognition* (Schantz 1982). Notons en outre que, dans un article proposant un panorama de l'évolution des technologies de l'information dans les années 1990, le développement des OCR est mentionné parmi les progrès importants (Bowers 2018).

Plus proche de nous, le projet de numérisation Google Books, entamé en 2004, a permis une grande reconnaissance de la technologie OCR et des possibilités qu'elle offre (Nagy 2016). En 2005, la mise à disposition du premier logiciel OCR libre, Tesseract (Smith 2007), ouvre la voie à une très large diffusion de cette technologie, maintenant accessible à tous (Blanke, Bryant, Hedges 2012).

Depuis, de nombreux projets de recherche dans ce domaine ont vu le jour, mais il est important de mentionner le plus influent au niveau européen, IMPACT. Ce projet à financement européen lancé en 2008 vise à proposer des outils et des méthodes de travail permettant l'océrisation de documents historiques numérisés avec un très haut niveau de précision (Balk, Ploeger 2009). L'aboutissement principal de ce projet est la création du centre de compétences IMPACT (IMPACT 2013) qui propose des outils, des lexiques et des numérisations pouvant servir de données d'entraînement.

2.1.2 Fonctionnement et perfectionnement

Avant l'océrisation, une numérisation de qualité est indispensable. Il est habituellement recommandé de choisir un format TIFF non compressé et de préparer les numérisations, en rognant les bords vides par exemple (Zhou 2010), ceci afin de simplifier le travail de l'OCR. En outre, il existe des méthodes d'évaluation automatique des numérisations, permettant d'assurer une précision optimale des OCR (Brener, Iyengar et Pianykh 2005).

L'océrisation à proprement parler se déroule en quatre phases : le prétraitement, ou *preprocessing*, la segmentation, ou *layout analysis*, la reconnaissance, ou *recognition*, et le post-traitement, ou *post-processing* (Blanke, Bryant, Hedges 2012). La première phase consiste essentiellement à binariser l'image pour distinguer les pixels 1 (les caractères) des pixels 0 (le fond), et à supprimer le bruit, afin de déterminer où se trouve le texte sur l'image. La seconde permet de repérer les lignes de textes et de délimiter les caractères. La troisième implique une extraction et une classification des *features* pour reconnaître lesdits caractères (Anugrah, Bintoro 2017), et la dernière phase consiste à corriger l'*output* pour diminuer le taux d'erreurs (Blanke, Bryant, Hedges 2012).

Pour cette dernière phase, plusieurs méthodes sont utilisées. Le *machine learning*, ou apprentissage supervisé, est rapidement apparu comme une solution efficace. Un article de 1992 précise déjà que l'usage du *machine learning* a permis aux auteurs de corriger 46% des erreurs d'océrisation avec un taux de précision de 91% sans intervention humaine (Sun et al. 1992).

De nos jours, le *machine learning* demeure l'une des solutions les plus recommandées pour la correction, souvent couplée avec l'utilisation de modèles de langue et de dictionnaires (Kissos, Dershowitz 2016). Ces deux procédés nécessitent néanmoins la présence de données d'entraînement fiables, c'est-à-dire des textes numérisés du même type et océrisés parfaitement, les *ground truth*. Certains chercheurs ont cependant tenté de proposer des méthodes de correction en l'absence de *ground truth* avec des résultats plutôt satisfaisants (Ghosh et al. 2016).

Des méthodes plus récentes proposent l'usage d'apprentissage statistique (Mei et al. 2018) ou encore de la distance de Levenshtein, qui permet de mesurer la différence entre des chaînes des caractères (Hládek et al. 2017). Un outil *open source* développé dans le cadre du projet IMPACT, PoCoTo, permet d'accélérer le repérage et la correction d'erreurs sur la base de modèles de langues adéquats (Vobl et al. 2014).

Des démarches moins techniques sont également en usage, tel le *crowdsourcing*, qui permet d'impliquer des volontaires pour corriger les erreurs de l'OCR (Clematide, Furrer, Volk 2016). Ce *crowdsourcing* peut prendre des formes diverses, à l'image des jeux créés par la *Biodiversity Heritage Library* et testés avec succès (Seidman et al. 2016).

2.1.3 Évaluation des performances

Dans un projet d'océrisation, il est essentiel d'évaluer les performances des logiciels OCR afin de sélectionner le mieux adapté et/ou d'optimiser celui sélectionné. Dans les années 1990 déjà, une équipe de l'*Information Science Research Institute* publiait annuellement les résultats de tests de précision des logiciels OCR disponibles sur le marché à l'époque. Leur méthode consistait à océriser un échantillon aléatoire de documents de différentes natures (journaux, textes de lois etc.) et en différentes langues afin d'estimer lequel présentait le moins d'erreurs.

La métrique principale utilisée dans leurs recherches est la précision des caractères, selon le calcul suivant : $\frac{\mathrm{n-nombre\ d'erreurs}}{\mathrm{n}}$, où n représente le nombre de caractères de l'*input* (Rice, Jenkins, Nartker 1996).

Cette métrique est encore utilisée de nos jours et enrichie. Un article de 2018 (Karpinski, Lohani, Belaïd 2018) propose en effet les calculs suivants : Erreurs = caractères ajoutés + caractères omis + caractères substitués. Caractères corrects = caractères de l'*output* – erreurs. Précision = caractères corrects / caractères dans l'*output* (*Hypothesis zone*). Rappel = caractères corrects / caractères dans l'i*nput* (*Reference zone*). Ces mêmes calculs peuvent être étendus aux mots entiers afin de déterminer si la segmentation a été effectuée correctement (Saber et al. 2016).

Ces métriques peuvent en outre être complétées par l'utilisation de la F-mesure, qui se calcule ainsi : F = 2x (précision x rappel) / (précision + rappel). Ceci permet en effet de prendre ces deux métriques en compte au sein d'un seul calcul (Bao, Zhu 2014). Un outil *open source*, ocrevaluation, a été développé pour permettre d'automatiser ces calculs sur la base des *ground truth* et des *outputs* proposés par l'OCR sélectionné (Carrasco 2014).

2.2 OCR - un champ de recherche en mouvement

2.2.1 Réseaux neuronaux artificiels

Parmi les avancées récentes dans le domaine de l'océrisation, l'utilisation de réseaux neuronaux artificiels est en pleine expansion. Le *machine learning*, au sens d'apprentissage supervisé, est déjà une pratique bien établie dans le domaine, mais l'usage d'apprentissage non-supervisé permet de nouveaux progrès.

En 2014 déjà, un article propose un état de l'art de l'usage des réseaux de neurones pour le prétraitement des documents avant océrisation (Rehman, Saba 2014). Cette technologie permet en effet de faciliter le repérage des lignes de textes, la segmentation et l'extraction de *features*, mais à l'époque de cet article, de nombreux problèmes se posent encore quant à ses limites, et au temps et à la masse de données d'entraînement nécessaires à son bon fonctionnement.

Un article de 2016 propose l'utilisation de la *back propagation* avec descente du gradient, qui permet de limiter la répercussion des erreurs. Dans cet article, les réseaux neuronaux sont essentiellement utilisés pour la classification et la reconnaissance des caractères à partir des pixels, et les résultats sont très positifs avec les caractères alphanumériques anglais. Les auteurs soulignent cependant que, dans le cas d'autres écritures, et entre autres celles présentant des ligatures entre les lettres, les résultats sont peu satisfaisants (Afroge, Ahmed, Mahmud 2016).

Dans un article de 2017 cité plus haut, des réseaux neuronaux sont également utilisés pour l'extraction de *features* et la classification, et les auteurs recommandent de procéder à une réduction du bruit au préalable pour réduire le temps d'entraînement du réseau neuronal et améliorer ses performances (Anugrah, Bintoro 2017).

Enfin, un article de 2019 présente un exemple d'ICR, pour *intelligent character recognition* – un OCR spécialement entraîné pour reconnaître les textes manuscrits – qui utilise un CNN, ou *convolutional neural network*, un type de réseau neuronal utilisé surtout pour la reconnaissance d'images non-textuelles. Cette technologie permet en effet de reconnaître une plus grande variété de caractères et de signes de ponctuation, d'après les auteurs (Ptucha et al. 2019).

2.2.2 Multilinguisme et systèmes d'écriture divers

L'un des domaines dans lesquels l'océrisation avance considérablement est la grande variété de langues et de formes d'écritures pour lesquelles un OCR peut proposer des résultats satisfaisants. Certaines langues non-européennes, comme le japonais ou le mandarin, ont très vite trouvé leur place au sein de la recherche dans ce domaine, du

fait du vaste marché possible. D'autres, moins répandues ou moins connues des chercheurs, ne reçoivent de l'attention que depuis peu.

C'est le cas, par exemple, des langues d'Inde, comme l'Odia (Dash, Puhan, Panda 2017), le Bangla, le Devanagari, le Tamoul etc. (Kumar et al. 2018). D'autres, comme l'arabe et le farsi, sont des objets de recherche depuis longtemps, mais nécessite encore du travail du fait de la complexité de leur système alphabétique et numérique (Amin Shayegan, Aghabozorgi 2014 ; Alghamdi, Teahan 2017). Il en est de même pour le finnois, dont la richesse des inflexions rend les résultats des OCR encore parfois hasardeux (Järvelin et al. 2016).

Ces langues ont cependant l'avantage d'être dotées d'une production écrite riche, offrant une abondance de données d'entraînement. D'autres, comme le yiddish et l'occitan, présentent une faible quantité de données disponibles. Dans ce type de cas, la création de lexiques et l'établissements de traits spécifiques des langues et des caractères en amont est conseillé, afin d'améliorer les résultats de l'apprentissage supervisé (Urieli, Vergez-Couret 2013).

2.2.3 Documents historiques et langues anciennes

2.2.3.1 Imprimés anciens

L'océrisation de documents historiques est l'un des champs de recherche phares dans le domaine. En effet, les imprimés anciens présentent de grandes variations quant aux typographies utilisées, et l'usage ou non de ligature ainsi que l'état général du document sont des éléments pouvant limiter les performances des OCR.

Dans un article de 2015, la Bibliothèque Nationale d'Autriche présentait ses projets « *Austrian Books Online* », « *Austrian Newspapers Online* » et « *Europeana Online* », des projets de numérisations et d'océrisation permettant la recherche plein-texte dans des documents historiques. Parmi les problèmes rencontrés, les auteurs notent l'utilisation contrainte de lexiques et modèles de langue modernes, mal adaptés à ce type de documents n'ayant pas de modèles de langues anciennes à disposition. Ils notent cependant que le projet IMPACT, mentionné plus haut, a entre autres permis de reconnaître l'importance de l'implémentation de lexiques et de modèles de langues anciennes adaptés. L'OCR seul ne peut pas tout faire (Kann, Hintersonnleitner 2015).

S'assurer que l'OCR est entraîné avec des données adéquates est donc indispensable, qu'il s'agisse de données linguistiques, au point de faire intervenir la technicité de la linguistique de corpus (Tumbe 2019), ou de signes d'écriture. En effet, certaines langues ont subi une évolution rapide de leur système d'écriture au cours de leur histoire. Un

article de 2016 présente le cas d'imprimés roumains produits entre le XVIIIème et le XXème siècle, dont l'écriture a beaucoup évolué, passant du cyrillique à différentes versions simplifiées de cet alphabet, puis enfin à l'écriture latine.

Là aussi, l'entraînement de l'OCR s'est fait à l'aide de données spécifiques – des lettres cyrilliques et latines roumaines des différentes époques concernées. Sans ces données, les performances du logiciel étaient fort limitées (Cojocaru et al. 2016).

Très récemment, un projet de l'Université de Würzburg en Allemagne a abouti à la création d'un logiciel libre, OCR4all, spécialement conçu pour traiter des imprimés anciens (Jost 2019). Cet outil fera partie de ceux que nous testerons dans nos recherches.

2.2.3.2 Manuscrits

La problématique des écritures se posent d'autant plus dans le cas de textes manuscrits, qui présentent de nombreuses difficultés pour les chercheurs dans le domaine de l'océrisation, et pour les humanités numériques en général.

Dominique Stutzmann, chargée de recherche à l'Institut de recherche et d'histoire des textes (IRHT), écrivait en 2017 que « [l]es années qui s'ouvrent sont certainement celles d'une interaction intense, aux bénéfices réciproques, entre l'homme et la machine en paléographie » (Stutzmann 2017), la paléographie étant l'étude et la transcription de manuscrits anciens.

En effet, beaucoup de chercheurs se penchent actuellement sur la question, et une compétition a même été organisée pour stimuler la recherche dans le domaine de la paléographie numérique. Un article de 2017 en retrace le déroulement, les méthodes développées dans ce cadre et les résultats, plutôt positifs (Kestermont, Christlein, Stutzmann 2017).

Un article plus récent encore se penche sur Transkribus, une plateforme libre de HTR, ou *handwritten text recognition*, et en démontre l'efficacité, dans le cas du corpus testé du moins (Muehlberger et al. 2019). La paléographie numérique a de beaux jours devant elle.

2.2.3.3 Langues anciennes : le cas du latin

Ne pouvant aborder le cas de toutes les langues anciennes, nous nous focaliserons sur le latin, et plus spécifiquement le latin de l'époque moderne, ou *Early Modern Latin*, car il s'agit de la langue qui nous concerne dans le cadre de ce projet de recherche.

Du fait de son corpus extrêmement riche, le latin est une langue ancienne qui a depuis longtemps intéressé les chercheurs dans le domaine de l'océrisation. En 2006 par

exemple, un article signale que les OCR de l'époque ne sont pas adaptés au traitement de cette langue, et propose l'implémentation de modèles de langue spécifiques, une solution déjà mentionnée plus haut (Reddy, Crane 2006).

Une problématique propre au latin, qui concerne également notre projet, est celle des abréviations. En effet, il est très fréquent de rencontrer des abréviations dans les textes latins, manuscrits comme imprimés. Par exemple, « dns » peut remplacer *dominus*, le seigneur, ou encore un tilde sur une voyelle signale généralement qu'elle est suivie d'un « m » ou d'un « n ». Un logiciel OCR ne peut pas *a priori* traiter ce type de cas. Pourtant, un article de 2003 propose déjà une solution, via un algorithme en trois temps, permettant de déterminer les résolutions possibles d'une abréviation et de sélectionner la meilleure en fonction du contexte (Rydberg-Cox 2003).

Actuellement, certains outils tentent de répondre à ces problèmes en se spécialisant dans le traitement de textes anciens, comme OCR4all mentionné plus haut, voire dans les textes latins, comme Latinocr.org, qui propose des jeux de données d'entraînement dont nous nous servirons dans nos recherches.

3. Tests de logiciels OCR

La première partie de notre projet consistait à tester différents logiciels d'océrisation *open source* afin de déterminer lequel offrait les meilleurs résultats sans post-correction. Ce chapitre présente cette phase du projet.

3.1 OCR sélectionnés

Pour des raisons de faisabilité, nous ne pouvions tester tous les logiciels OCR gratuits et *open source* disponibles, et nous avons donc porté notre choix sur quatre d'entre eux. Nous avons sélectionné Tesseract et OCR4all, car ce sont ceux que la littérature récente mentionne le plus, et Kraken et Calamari, car ce sont les *forks* les plus à jour d'OCRopus, un projet également très présent dans la littérature.

3.1.1 Tesseract

Tesseract est un logiciel d'océrisation développé initialement par Hewlett Packard entre 1984 et 1994, puis rendu *open source* en 2005 (Smith 2007). Il a ensuite été repris en 2006 par Google, qui en assure depuis la maintenance et l'a mis à disposition sous la licence Apache-2.0 sur github.com/tesseract-ocr. Il a pour avantage de proposer des modèles pré-entraînés dans de nombreuses langues, avec la possibilité de combiner les modèles entre eux. Il autorise en outre la création de modèles sur la base de numérisations.

3.1.2 Kraken

Kraken est un *fork* du projet OCRopus, maintenant nommé OCRopy, lancé en 2007 par Thomas Breuel, du *Deutsches Forschungszentrum für Künstliche Intelligenz*, avec le soutien de Google (Breuel 2007). Kraken est supposé rectifier certains problèmes que posent OCRopus, mais présente des fonctionnalités similaires. Comme Tesseract, il propose quelques modèles pré-entraînés et offre la possibilité d'en entraîner soi-même. Il est développé en Python, conçu pour être utilisé sur Linux, et a son site dédié : kraken.re.

3.1.3 Calamari

Le logiciel d'océrisation Calamari, lancé en 2018, est basé sur les projets OCRopy et Kraken. Il est également implémenté en Python et utilise des réseaux neuronaux artificiels pour optimiser ses résultats (Wick, Reul, Puppe 2018). Il est disponible en ligne sur github.com/Calamari-OCR.

3.1.4 OCR4all

OCR4all est un projet de l'Université de Würzburg en Allemagne lancé en 2019. Il a été conçu pour traiter les documents historiques et est doté d'une interface qui facilite son utilisation, sans que des connaissances en informatiques préalables soient nécessaires (Jost 2019). Le projet, qui intègre déjà différents logiciels, tels que Calamari et Kraken, est en cours d'intégration de Tesseract pour la reconnaissance de caractères. Il est à disposition du public sur github.com/OCR4all.

3.2 Méthodologie

3.2.1 Données d'entraînement

Notre jeu de données étant composé de 29 livres numérisés contenant plus d'une centaine de pages chacun, nous avons choisi de sélectionner 29 images comme données d'entraînement, chacune extraite de l'un des 29 livres. Cette sélection, faite au hasard à l'aide d'un script Python nous a permis d'obtenir un échantillon de chacun des livres, ceux-ci pouvant présenter des variantes au niveau de la typographie.

Nous avons ensuite transcrit manuellement ces numérisations dans des fichiers textes afin d'obtenir le *ground truth*, c'est-à-dire la « réalité », l'objectif à atteindre pour l'OCR. Ceci nous permet de mesurer les performances des différents logiciels que nous devons tester.

Nous avons en outre transcrit ces numérisations de deux manières différentes : une première transcription dite « diplomatique », au plus proche du document, et donc au plus proche des résultats qu'un logiciel OCR devrait pouvoir obtenir, et une seconde transcription dite « normalisée », qui servira de base à la post-correction.

Dans cette dernière les abréviations ont été résolues et des choix ont été faits pour simplifier la recherche et la lecture du texte. Le caractère æ a été remplacé par ae, les i et les j ont tous été remplacés par des i et les u et les v ont tous été remplacés par des u, sauf lorsqu'il s'agissait de chiffres romains. Ces choix ont été faits sur la base d'habitudes de recherches en latin et de règles d'orthographie usuelle de cette langue.

3.2.2 Logiciels et paramétrage

Pour chacun des logiciels d'océrisation choisis, nous avons effectué des tests en trois phases différentes.

La première phase consistait à tester les différents logiciels avec leur modèle standard, leurs paramètres par défaut et sans aucune modification de notre part. Ceci nous a

permis, sur un premier test, de voir quel logiciel était le plus performant, avec ses réglages de base. Les résultats sortis étaient alors totalement bruts.

Nous nous sommes autorisés, lors de la seconde phase, à tester d'autres modèles et à modifier les réglages que proposent chaque logiciel. Ceci nous a ainsi permis de comparer les performances de chaque logiciel avec différents paramètres. Les résultats sortis ont été comparés à ceux de la phase précédente, afin de pouvoir déterminer quels paramètres avaient une influence positive sur nos premiers résultats.

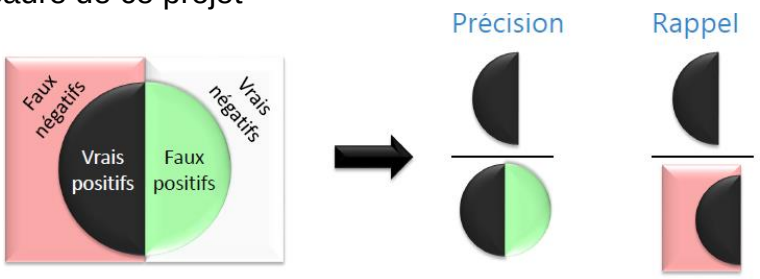
Lors de la troisième et dernière phase de l'évaluation des différents logiciels, nous avons effectué un pré-traitement sur les images de notre jeu de test, afin de les retravailler et de voir si cela permettait d'optimiser nos résultats.

Toute cette phase d'évaluation nous a permis de comparer les modèles d'apprentissage et la qualité des *outputs* de chacun des logiciels, afin de sélectionner le plus performant. Ces océrisations et calculs de résultats pouvant être longs, nous avons créé un script de *threading* permettant de lancer l'océrisation de plusieurs images en parallèle, afin de gagner du temps.

3.2.3 Métriques

Pour mesurer la performance des logiciels testés, nous avons utilisé des métriques usuelles dans le domaine des Sciences de l'Information, à savoir la précision et le rappel (Burgy, Gerson, Schüpbach 2020). La précision représente ici le pourcentage de caractères corrects trouvés sur la totalité des caractères de l'*output*. Le rappel représente ici le pourcentage de caractères corrects trouvés sur la totalité de caractères de l'*input*. La figure ci-dessous illustre ces deux métriques.

Figure 1 : représentation de la précision et du rappel extraite d'un poster scientifique réalisé par nos soins dans le cadre de ce projet



Pour permettre de faire la balance entre précision et rappel, nous avons également utilisé la moyenne harmonique de ces métriques, nommée « F-mesure », ou F_1, qui se calcule selon cette formule :

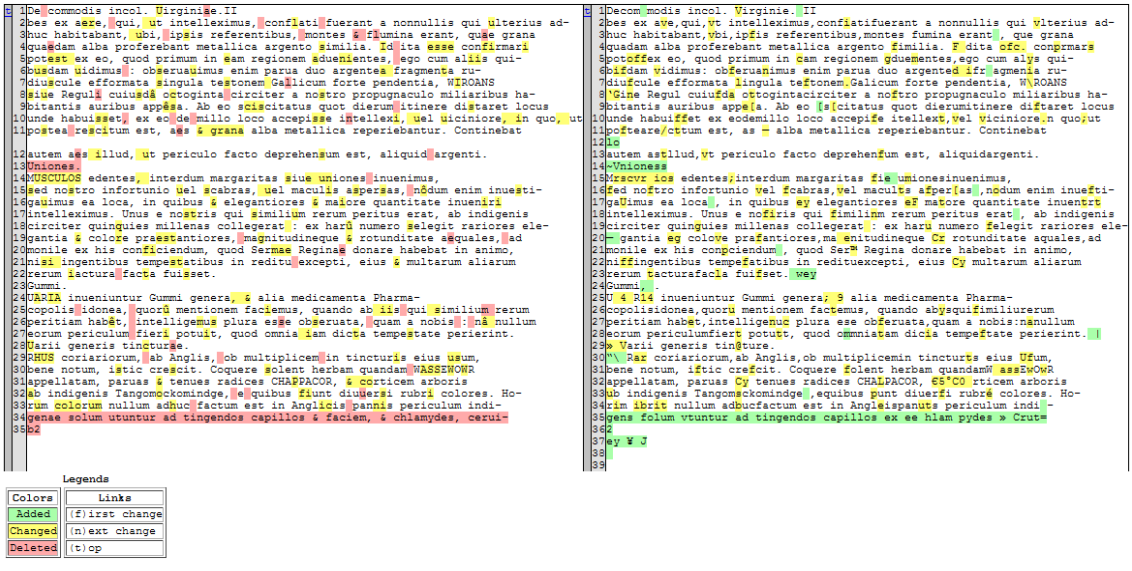
$F = \frac{2PR}{P+R}$ où P est la précision et R le rappel (Sasaki 2007).

Comme il s'agit de texte, nous avons utilisé ces métriques à la fois à l'échelle des caractères et à l'échelle des mots (Burgy, Gerson, Schüpbach 2020), ceci afin de pouvoir mieux décider quelles stratégies choisir pour l'entraînement des modèles et la post-correction, en vue d'optimiser nos résultats. De manière générale, il est fréquent que les résultats au niveau des mots soient moins bons qu'au niveau des caractères, car il suffit qu'un caractère soit incorrect pour que le mot entier soit considéré comme faux.

Nous avons également utilisé la distance de Levenshtein, qui permet de comparer deux chaînes de caractères et de repérer le nombre de caractères ajoutés, supprimés ou substitués. L'algorithme qui effectue cette opération donne en sortie une somme des erreurs (arvindpdmn, 2019), ce qui nous a été utile dans le calcul des métriques précédentes.

Pour obtenir un retour visuel, nous avons utilisé la librairie « difflib » qui affiche les caractères ajoutés, supprimés et substitués. Cela permet de vérifier quelles parties des *outputs* présentent des erreurs.

Figure 2 : comparaison d'une transcription (gauche) avec un output de Tesseract (droite) grâce à la librairie "difflib"



Nous avons également testé un outil permettant le calcul des différentes métriques et produisant un fichier semblable celui de la librairie « difflib », à savoir ocrevalUAtion, disponible sur github.com/impactcentre/ocrevalUAtion. Nous avons finalement décidé de ne pas l'utiliser car nous avions déjà créé nos propres scripts de calcul des résultats

qui nous permettait d'ajouter ou de modifier des fonctionnalités en fonction des besoins, ce que ne permet pas cet outil.

3.3 Résultats

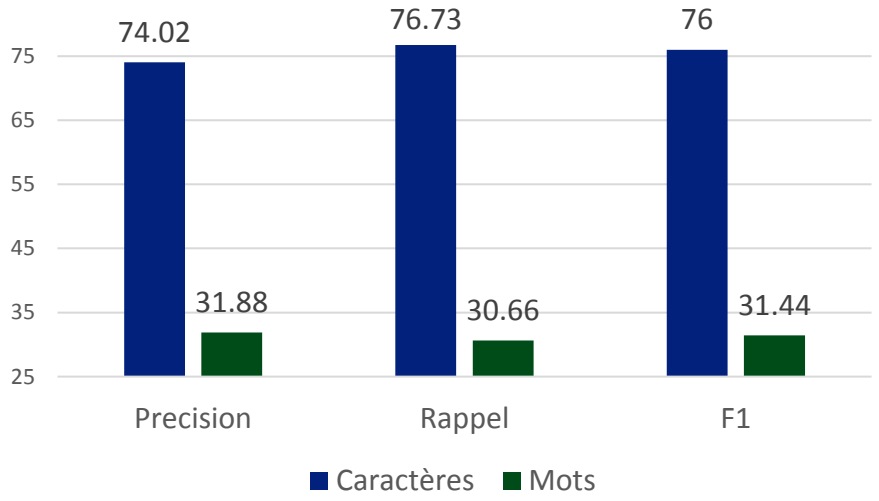
3.3.1 Tesseract

3.3.1.1 Tesseract – phase 1

Dans la première phase de test avec Tesseract, nous avons utilisé le logiciel avec son modèle standard anglais et ses paramètres par défaut. Les premiers résultats sont déjà encourageants au niveau des caractères avec une F_1 de 76%.

Les résultats au niveau des mots sont en revanche bien plus faibles. Nous avons obtenu une F_1 de 31,4%, ce qui peut s'expliquer par la problématique mentionnée dans la partie Métriques.

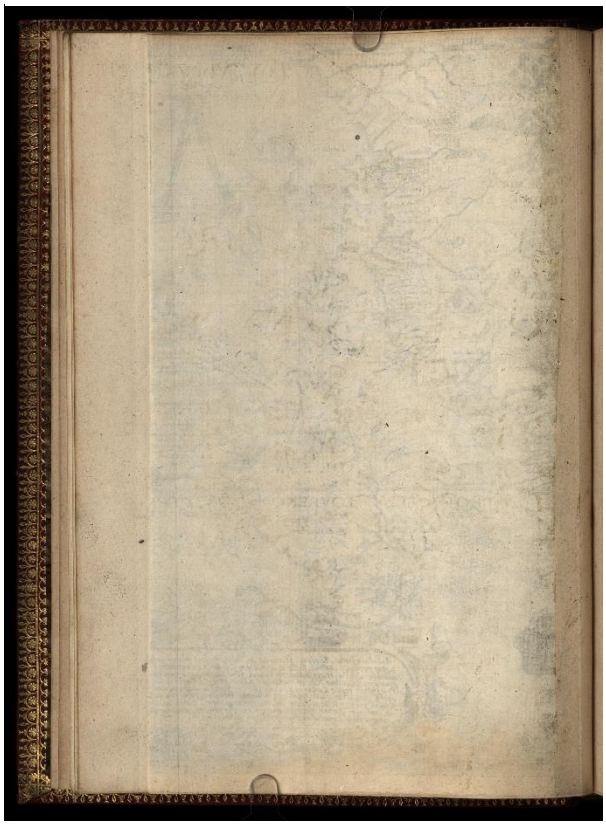
Figure 3 : Tesseract – phase 1 – modèle anglais

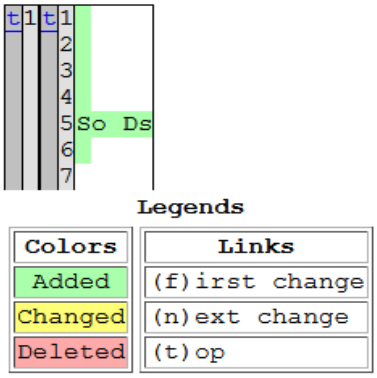


Nous avons cependant repéré quelques problèmes. Dans le cas de pages vides, par exemple, notre script Python de calcul automatique des métriques ne parvient pas à comparer les deux fichiers – la transcription et l'*output* – et produit des résultats incohérents. Afin de calculer les différentes métriques, nous avons en effet besoin de connaître le nombre de caractères corrects ainsi que le nombre de caractères du *ground truth* (la transcription).

Dans le cas précis d'une transcription ne comportant pas de texte, le nombre de caractères corrects et le nombre de caractères de la transcription seront toujours égaux à 0 et fourniront toujours un résultat de 0 (ou une erreur de division par 0). Nous avons donc décidé de définir automatiquement les valeurs de la précision, du rappel et de la F_1 dans ce cas précis.

Figure 4 : numérisation d'une page blanche de la collection de Bry et *output* de Tesseract





Dans le cas ci-dessus, la transcription contient 0 caractères. L'OCR, lui, a trouvé 10 caractères (espaces compris), tous faux. Les calculs des différentes métriques seront alors les suivants :

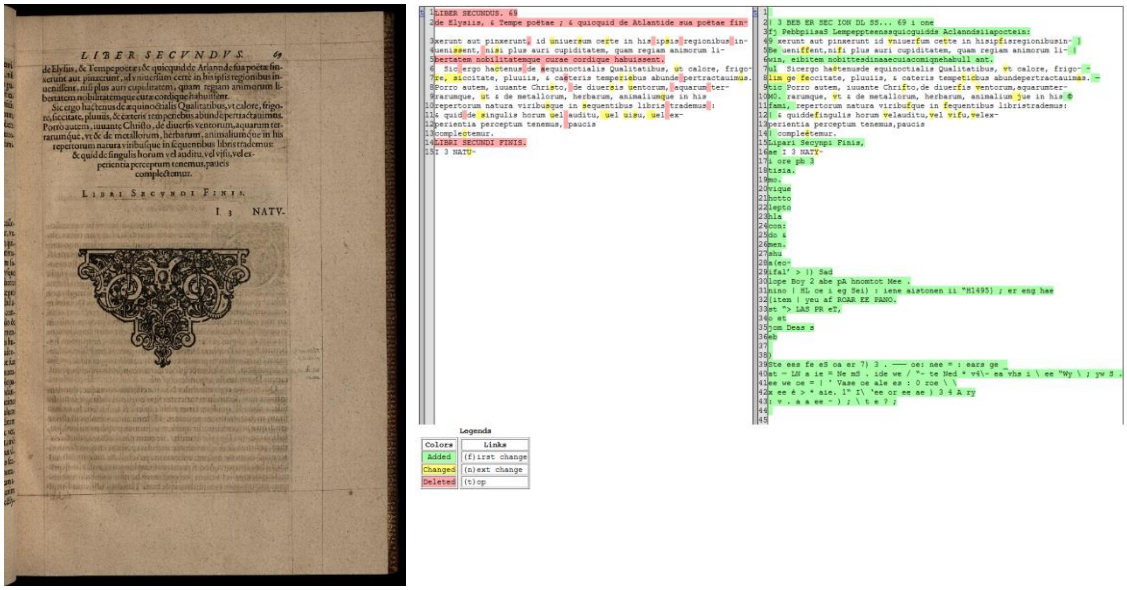
$$P = \frac{0}{10} = 0$$

$$R = \frac{0}{0} \rightarrow \text{Division par 0 !}$$

Dans tous les cas, les résultats des métriques donneront 0 ou une division par 0. Donc en définissant les résultats à 0, nous limitons les risques de résultats incalculables mais ne faussons pas ces derniers.

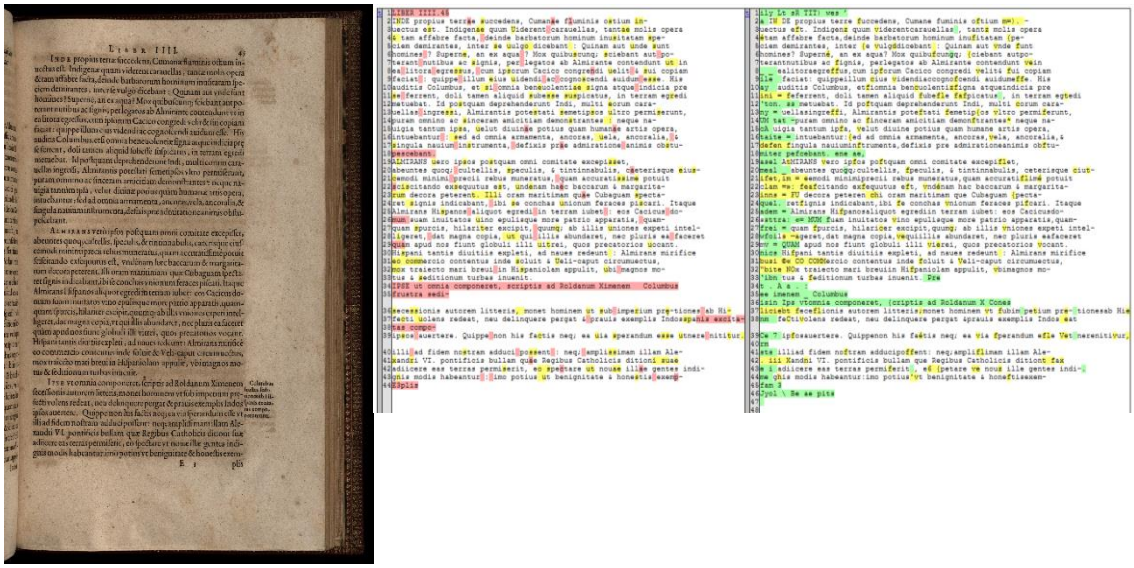
Dans le cas de numérisations comportant des typographies différentes ou des gravures, les résultats tendent à chuter, car le logiciel peine à les traiter et cherche des caractères là où il n'y en a pas. Dans l'exemple ci-dessous, on peut voir que Tesseract a « trouvé » des caractères dans la gravure.

Figure 5 : numérisation d'une page de la collection de Bry comportant une image ; transcription et *output* de Tesseract



Enfin, nous avons pu observer que, dans la plupart des numérisations, une partie du texte de la page adjacente est visible. Tesseract tend à océriser ces caractères également, ce qui influe sur la précision. La figure ci-dessous illustre bien cette problématique, car on voit de nombreux caractères ajoutés – surlignés en vert – au début de la plupart des lignes de l'*output*.

Figure 6 : numérisation d'une page de la collection de Bry dont la page adjacente est visible ; transcription et *output* de Tesseract.

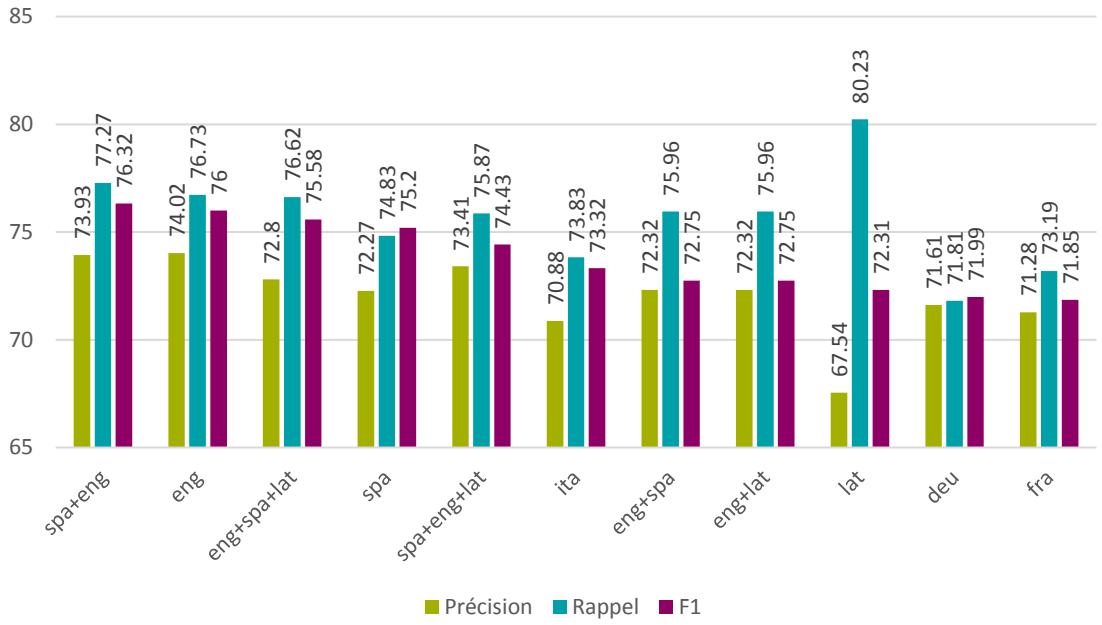


Ces observations nous ont été très utiles pour les phases suivantes.

3.3.1.2 Tesseract – phase 2

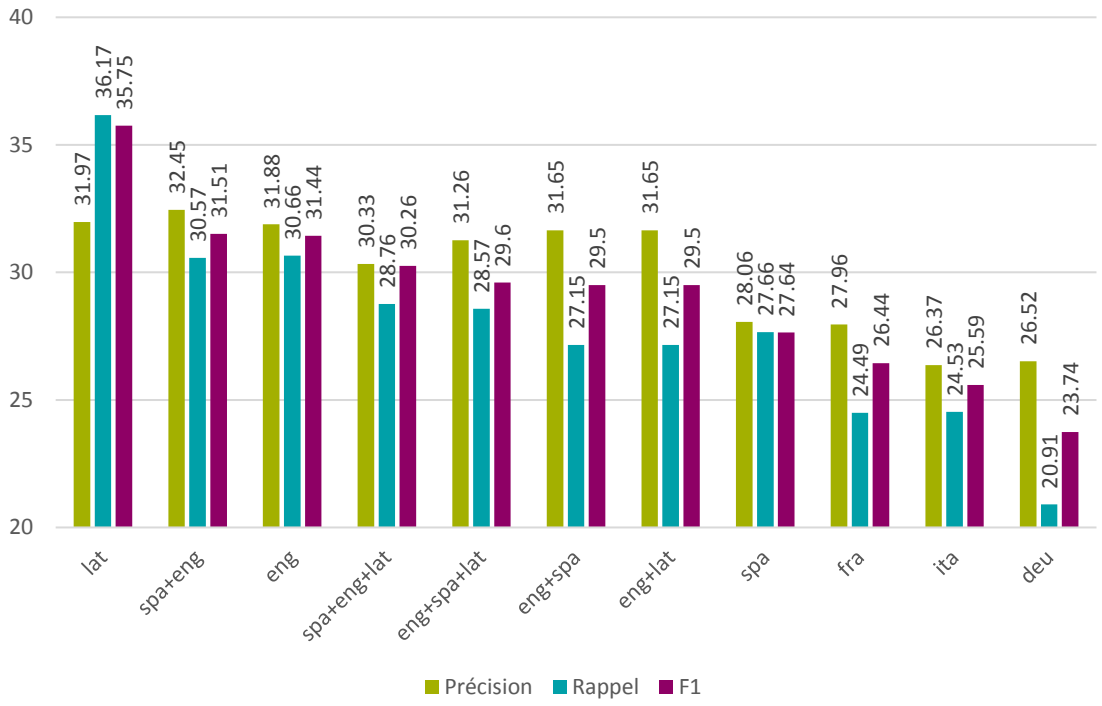
Dans la seconde phase, nous avons testé différents modèles proposés par Tesseract, en sélectionnant des langues relativement proches du latin, à savoir les modèles allemand (deu), anglais (eng), espagnol (spa), français (fra), italien (ita), et latin (lat) bien entendu. Nous avons également essayé de combiner plusieurs de ces modèles entre eux. Les résultats au niveau des caractères montrent que le modèle anglais testé dans la phase 1 se fait légèrement dépasser par la combinaison des modèles espagnol et anglais avec une F_1 de 76.32%.

Figure 7 : Tesseract – phase 2 – modèles de langues – caractères



Au niveau des mots, nous pouvons voir que le meilleur modèle est le modèle latin avec une F_1 de 35.75%, suivi par la combinaison des modèles espagnol et anglais avec une F_1 de 31.51%.

Figure 8 : Tesseract – phase 2 – modèles de langues – mots



On voit ici que le choix de l'ordre des modèles dans une combinaison est crucial. En effet, la combinaison espagnol/anglais offre de bien meilleurs résultats que la combinaison anglais/espagnol.

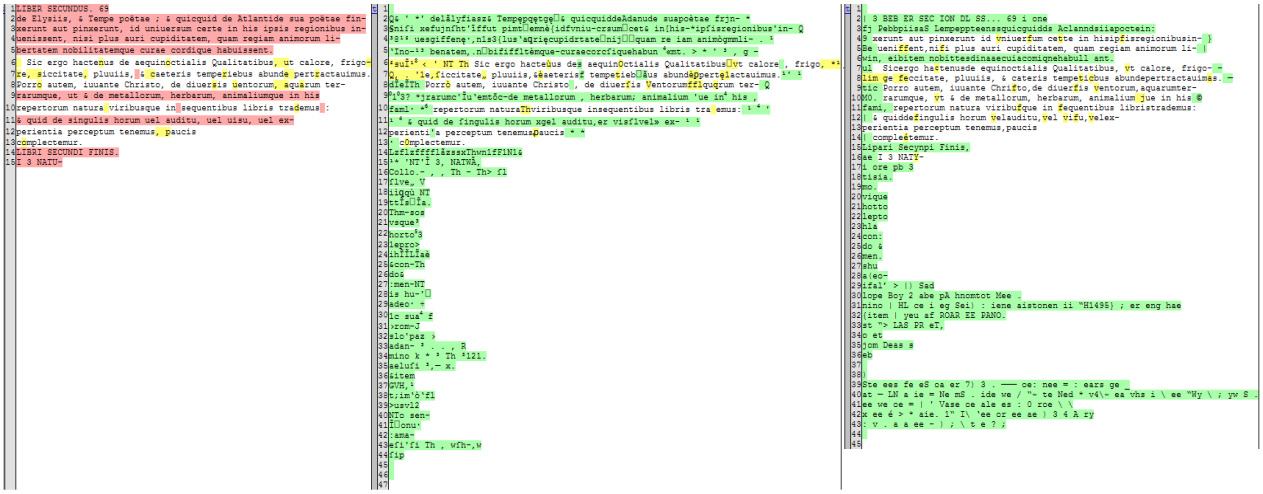
Il est également intéressant de noter que, bien que la F_1 de la combinaison espagnol/anglais soit la plus élevée, ce n'est pas forcément le cas pour la précision et le rappel. On peut voir ici que le modèle avec la précision la plus élevée est le modèle anglais tandis que le modèle avec le rappel le plus élevé est le latin.

De fait, il est frappant de voir que le modèle latin, pourtant la langue de cette édition de la collection de Bry, a un taux de rappel si élevé, alors que sa précision est la plus mauvaise de tous les modèles testés. En effet, le modèle latin gère bien moins bien les espaces blancs et les marges ainsi que les images, et ajoute beaucoup de caractères incorrects.

En outre, le modèle latin donne des résultats moins bons lorsqu'il est confronté à des typographies différentes. Cela peut s'expliquer par la manière dont ces différents modèles sont entraînés. En effet, certaines langues vivantes, comme l'anglais, permettent de créer de vastes jeux de données d'entraînement présentant des typographies variées, alors que, pour le latin, la quantité de données à disposition est moindre, et les résultats s'en ressentent (theraysmith, 2017).

Ces deux problématiques sont visibles dans l'exemple ci-dessous.

Figure 9 : comparaison d'une transcription avec un *output* modèle latin (centre) et modèle anglais (droite)



Dans cette phase, nous avons également testé différents paramètres de segmentation de pages (ou *psm*) qu'implémente Tesseract :

0 = *Orientation and script detection (OSD) only*.

1 = *Automatic page segmentation with OSD*.

2 = *Automatic page segmentation, but no OSD, or OCR*

3 = *Fully automatic page segmentation, but no OSD. (Default)*

4 = *Assume a single column of text of variable sizes.*

5 = *Assume a single uniform block of vertically aligned text.*

6 = *Assume a single uniform block of text*.

7 = *Treat the image as a single text line.*

8 = *Treat the image as a single word*.

9 = *Treat the image as a single word in a circle*.

10 = *Treat the image as a single character*.

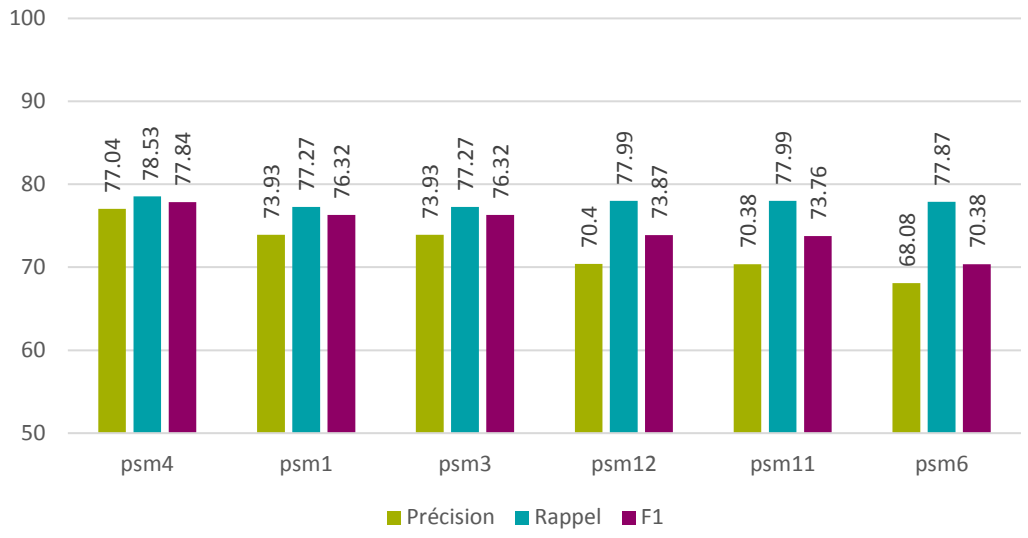
11 = *Sparse text. Find as much text as possible in no particular order.*

12 = *Sparse text with OSD*.

13 = *Raw line. Treat the image as a single text line*

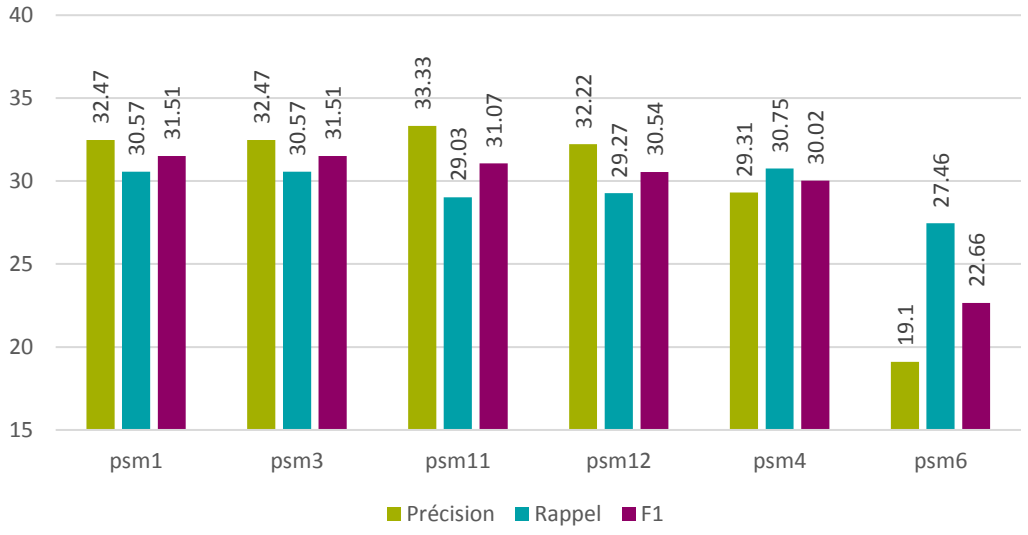
Le graphique ci-dessous présente les résultats obtenus au niveau des caractères avec les six paramètres de segmentation les plus performants et le modèle anglais. On voit bien que la meilleure segmentation pour notre problème est la numéro 4 avec une F_1 de 77,84%. La 1 et la 3 viennent ensuite, avec toutes les deux une F_1 de 76,32%.

Figure 10 : Tesseract – phase 2 – *psm* – caractères



Au niveau des mots, les meilleures segmentations sont la 1 et la 3, avec une F_1 de 31,51%.

Figure 11 : Tesseract – phase 2 – *psm* – mots



Ces résultats ont permis d'aiguiller nos décisions pour la dernière phase.

3.3.1.3 Tesseract – phase 3

Pour la dernière phase, nous avons utilisé les meilleurs paramètres identifiés précédemment (modèle espagnol/anglais et segmentation 1 ou 4) afin de vérifier si un pré-traitement sur les images permet d'améliorer les résultats.

Nous avons essayé 3 types de pré-traitements différents :

- Modification de l'image en nuance de gris (threshold)
- Modification de la taille de l'image (resample)
- Modification de la taille et modification en nuance de gris (full)

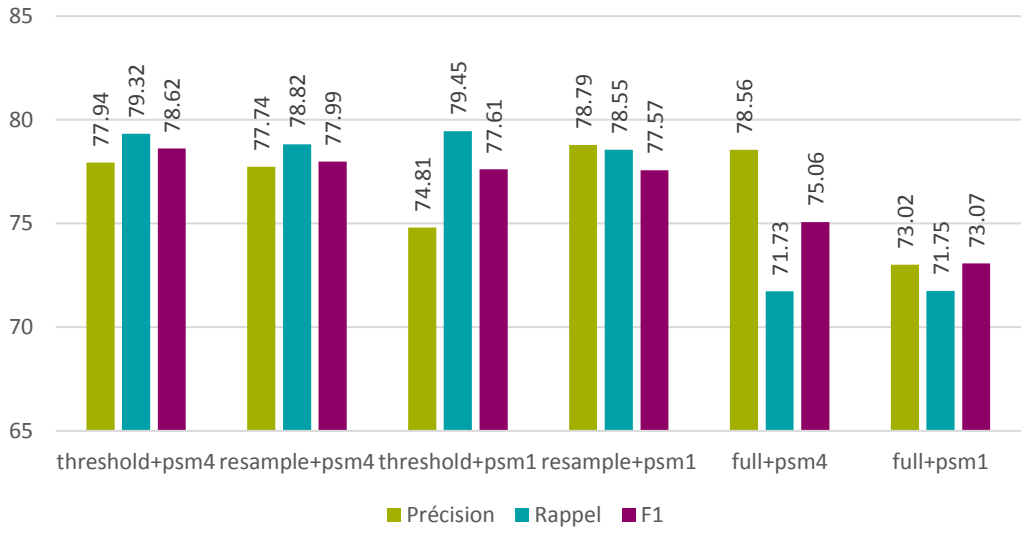
La méthode *threshold* consiste à modifier tous les pixels dépassant un certain seuil de couleur. Tout pixel plus clair que le seuil donné, 20% dans notre cas, sera automatiquement transformé en pixel blanc. Tout autre pixel sera modifié en pixel noir.

La méthode *resample* consiste à rogner l'image en fonction de sa position dans le livre. Si la page est un recto (page de droite dans un livre), 400 pixels sont rognés de la gauche de l'image et 200 de la droite, et inversement si la page est un verso. Cela permet de supprimer assez facilement les pages adjacentes visibles. Néanmoins, comme les valeurs sont fixes, il est possible que le script rogne trop et qu'une partie du texte soit perdu.

La méthode *full* utilise les deux méthodes ci-dessus. Pour chacune de ces trois méthodes, nous avons également ajouté une bordure blanche sur l'image, comme la documentation de Tesseract le conseille (Cimon 2019). Toutes les modifications ont été faites à l'aide de la librairie « ImageMagick », disponible sur imagemagick.org.

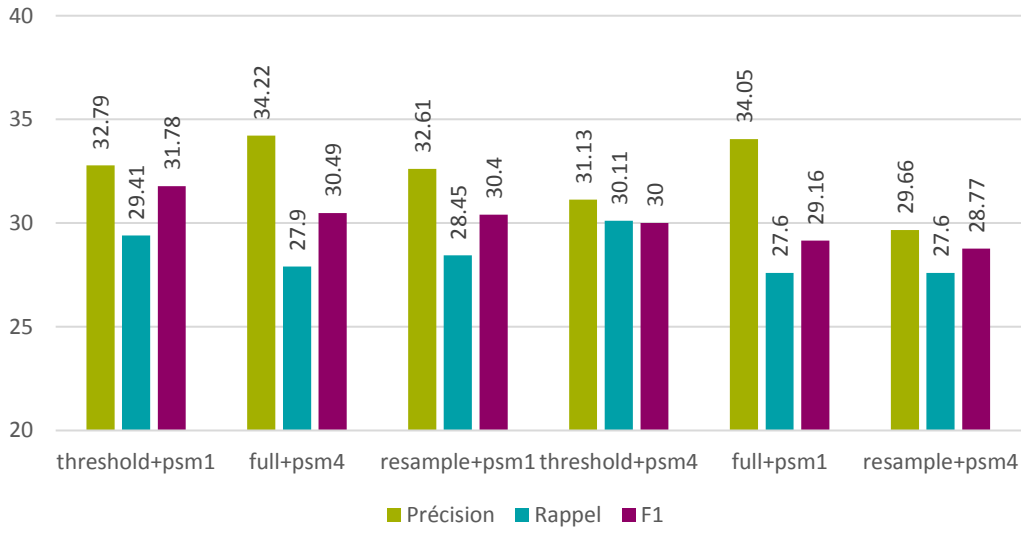
Les résultats obtenus montrent que ces méthodes peuvent améliorer la qualité des outputs. En effet la précision au niveau des caractères augmente de 0.9%, le rappel de 0.79% et la F_1 de 0.78%.

Figure 12 : Tesseract – phase 3 – modèle « spa+eng » – pré-traitement des images – caractères



Au niveau des mots, une légère amélioration des résultats est aussi visible. Nous gagnons 0.28% en précision, perdons 1.16% en rappel et gagnons 0.27% en F_1.

Figure 13 : Tesseract – phase 3 – modèle « spa+eng » – pré-traitement des images – mots



Ces trois méthodes de pré-traitement sont encore naïves et méritent d'être améliorées, mais nous pouvons d'ores et déjà affirmer que le pré-traitement des images augmente effectivement les résultats.

En définitive, on peut dire que la combinaison des modèles espagnol et anglais avec une segmentation de type 1 ou 4 semble être la méthode la plus adaptée à notre

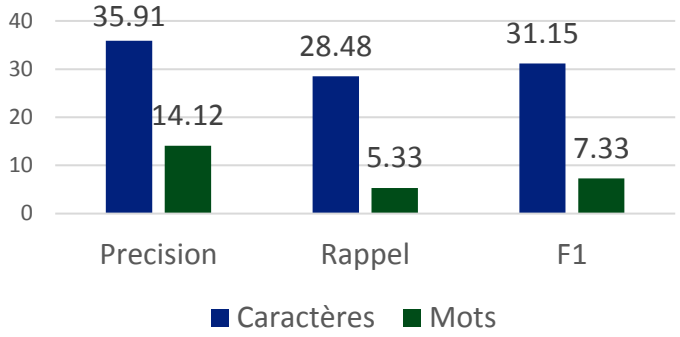
problème. Un pré-traitement sur les images augmente également légèrement les résultats. À ce stade, nous pouvons compter sur une F_1 de **78.62%** au niveau des caractères et de **31.78%** au niveau des mots. Notons également que Tesseract est un logiciel stable, robuste et facile à utiliser qui ne nous a posé aucun problème de prise en main, ce qui est un avantage non négligeable.

3.3.2 Kraken

3.3.2.1 Kraken – phase 1

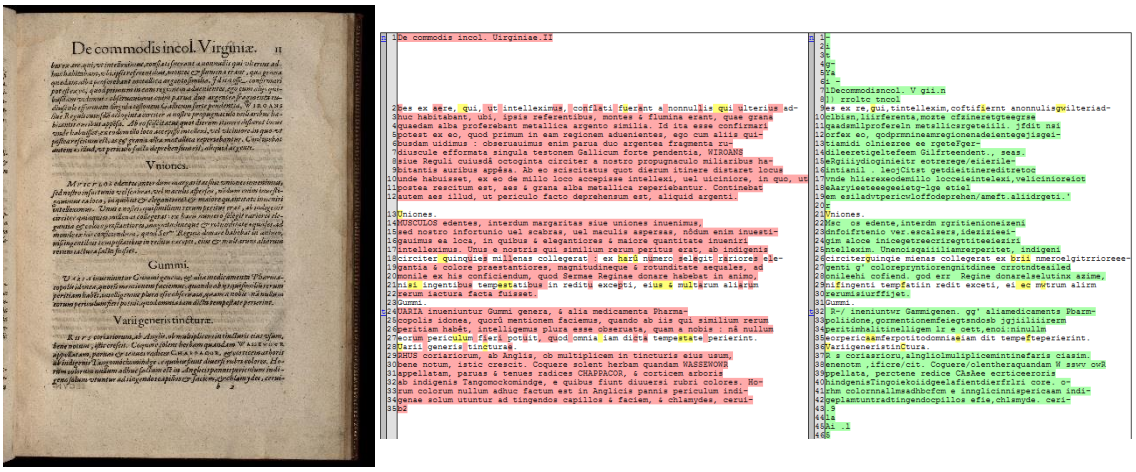
Dans la première phase, nous avons testé Kraken avec ses paramètres de base et son modèle standard anglais. Les premiers résultats sont peu concluants, surtout en comparaison de ceux obtenus avec Tesseract. En effet, on obtient une F_1 de 31.15% pour les caractères et 7.33% pour les mots.

Figure 14 : Kraken – phase 1 – modèle anglais



Kraken rencontre des problèmes similaires à ceux de Tesseract présentés plus haut. Cependant la fonction de pré-traitement intégrée à Kraken est bien moins performante et ne supprime pas suffisamment de bruit. Kraken a également du mal à découper les phrases en mots lorsque les espaces sont trop petits, faisant chuter les résultats.

Figure 15 : numérisation d'une page de la collection de Bry ; transcription et *output* de Kraken comportant beaucoup d'erreurs



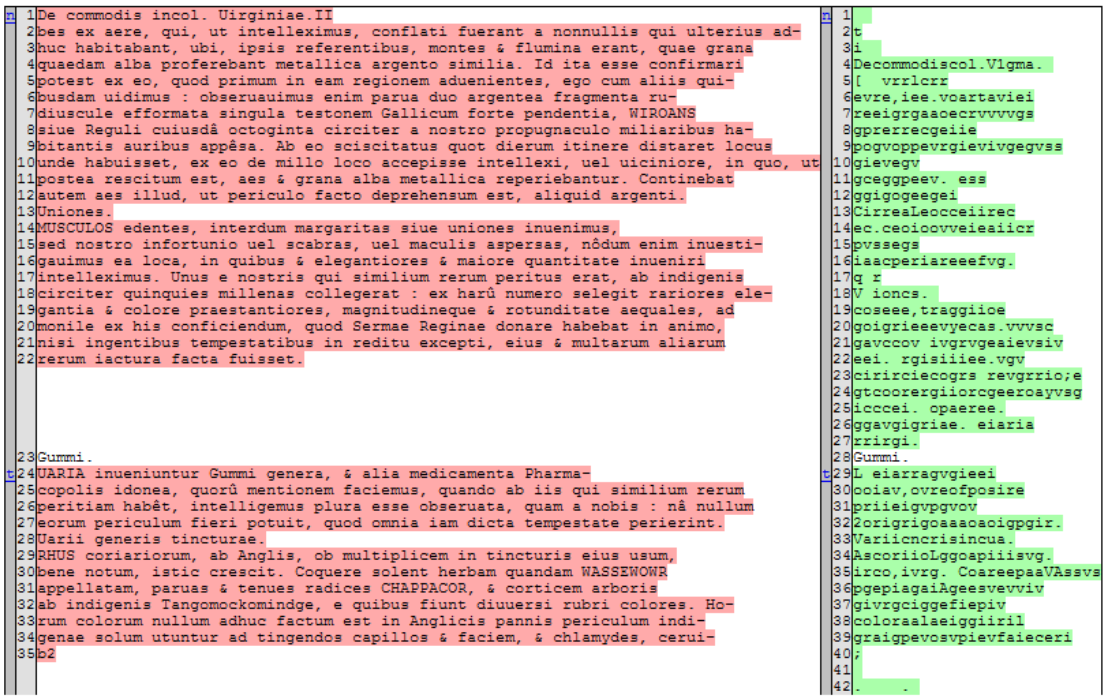
3.3.2.2 Kraken – phase 2

Dans la deuxième phase de test, nous avons cherché d'autres modèles pré-entraînés pour le logiciel Kraken. Malheureusement, très peu de modèles sont disponibles en ligne et nous en avons trouvé uniquement un d'exploitable : latin-teubner.

Nous l'avons utilisé pour le comparer avec le modèle anglais, mais il n'augmente pas les résultats, bien au contraire. La précision et le rappel sont catastrophiques. L'océrisation comporte tellement de caractères faux que le calcul de ces métriques donne des chiffres extrêmement proches de 0%. De ce fait, nous avons choisi de ne pas mettre de graphique de données pour cette phase.

La figure ci-dessous présente la transcription et l'*output* de Kraken pour la même numérisation que celle illustrée dans la phase précédente et permet ainsi de voir à quel point le modèle latin-teubner ne parvient pas à traiter notre *input*.

Figure 16 : transcription et *output* de Kraken modèle latin-teubner



Nous n'avons donc pas réussi à obtenir de résultats concluants avec Kraken. Il aurait sans doute fallu passer beaucoup plus de temps à tester différents modèles et paramètres, car Kraken est un logiciel reconnu dans le domaine de l'océrisation, et il n'est pas normal que nous ne puissions obtenir de meilleurs résultats.

Cependant, au vu du temps dont nous disposions pour la suite du projet, nous avons choisi d'arrêter ici les tests de Kraken, et de ne pas réaliser la troisième phase. Celle-ci

visait en effet à recadrer l'image, ce qui n'aurait pas résolu le problème de la quantité phénoménale d'erreurs au niveau des caractères et des mots.

Cette décision s'est également basée sur le fait que le plus gros problème de Kraken réside dans sa conception. En effet, confronté à une page blanche sans texte, le logiciel va tenter de segmenter le document sans succès. Il essaye à tout prix de trouver du texte et finit par tourner en boucle infinie sans donner de message d'erreur. De ce fait, nous avons dû enlever les images sans texte de notre jeu de test pour pouvoir obtenir les résultats ci-dessus. Ce défaut rendant les tests encore plus chronophages, nous avons jugé bon de passer au logiciel suivant.

3.3.3 Calamari

3.3.3.1 Calamari – phase 1

Pour tester Calamari, nous avons tout d'abord utilisé ses paramètres de base et un modèle standard, antiqua_historical. Rencontrant quelques problèmes techniques, nous avons alors testé d'autres modèles standards, à savoir antiqua_modern, fraktur_historical et fraktur_19th_century. Ces modèles sont basés sur deux catégories de typographie (et non de langues) : l'Antiqua, qui regroupe toutes les polices de caractères dérivées de l'écriture latine de l'Empire romain utilisées depuis la Renaissance en Occident (Beinert 2018), et la Fraktur, dérivée de la calligraphie gothique médiévale et utilisées essentiellement dans les régions de langue germanique jusque dans les années 1960 (Beinert 2019).

À notre grande surprise, tous les *outputs* que nous obtenions étaient vides, sans pour autant que nous ayons reçu de message d'erreur, et ce quel que soit le modèle. Nous avons alors choisi de passer à la deuxième phase malgré tout.

3.3.3.2 Calamari – phase 2

Face à l'échec de la première phase, nous avons essayé différents paramétrages avec chacun des modèles, sans succès. Les *outputs* restant désespérément vides, nous avons décidé d'arrêter les tests.

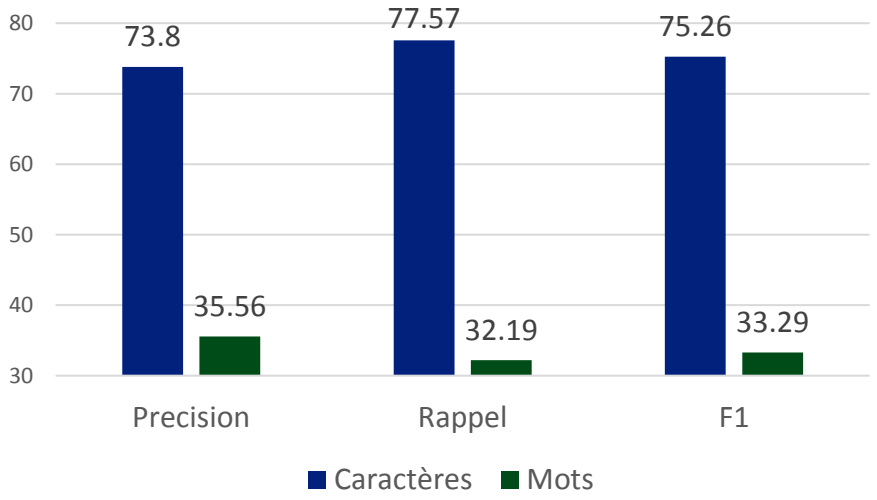
Tout comme pour Kraken, il aurait sans aucun doute valu la peine de continuer à faire des essais pour comprendre d'où venait le problème, car Calamari est également un logiciel reconnu dans le monde de l'océrisation, et nous aurions sans doute pu obtenir des résultats corrects, à terme. Cependant, dans le cadre de ce projet de recherche, la tâche eût été trop chronophage, et nous nous sommes donc résignés à passer au logiciel suivant.

3.3.4 OCR4all

3.3.4.1 OCR4all – phase 1

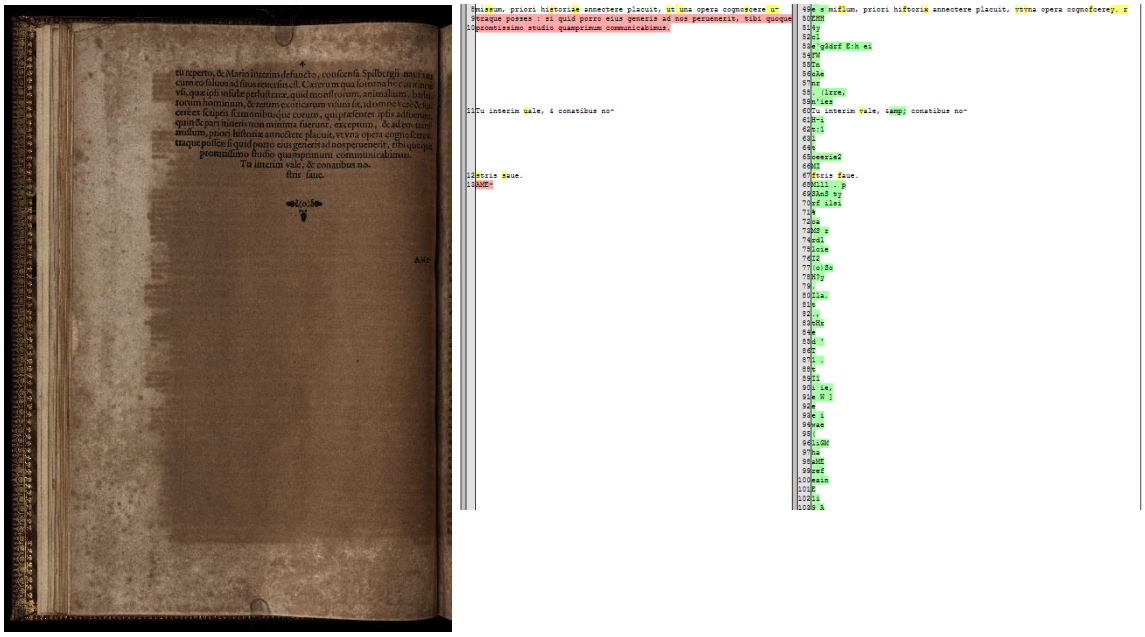
Tout comme pour les OCR précédemment testés nous avons utilisé les paramètres standards et un modèle standard d'OCR4all, antiqua_modern, pour la première phase de test. Les premiers résultats sont prometteurs avec une F_1 de 75.26% au niveau des caractères, donc légèrement plus élevée que celle de Tesseract dans la même phase. Les résultats au niveau des mots sont eux aussi encourageants avec une F_1 de 33.29%.

Figure 17 : OCR4all – phase 1 – modèle antiqua_modern



À l'instar de Tesseract, OCR4all peine à traiter les pages comportant des gravures, comme l'illustre l'image ci-dessous.

Figure 18 : numérisation d'une page de la collection de Bry ; extrait de la transcription et de l'*output* OCR4all



En outre, comme Kraken, il tend à chercher des caractères dans les espaces blancs, tournant en boucle infinie lorsqu'il est confronté à une page blanche, sans donner de messages d'erreur. Ce problème a été signalé sur leur dépôt github officiel et devrait être résolu à terme (Reul 2020).

Afin de pouvoir passer à la phase suivante et de pouvoir tester correctement les autres modèles d'OCR4all, nous avons dû supprimer les numérisations de pages blanches présentes dans notre jeu de données de test. Ceci peut fausser légèrement les résultats, car les calculs ne prennent plus en compte que 27 numérisations au lieu de 29.

3.3.4.2 OCR4all – phase 2

Dans la phase suivante, nous avons testé d'autres modèles que propose OCR4all, à savoir antiqua_historical, fraktur_historical et fraktur_19th_century. Le meilleur de tous les modèles est antiqua_historical, avec une F_1 de 83.12% pour les caractères et de 49.71% pour les mots. Ces résultats sont excellents !

Figure 19 : OCR4all – phase 2 – modèles de typographie – caractères

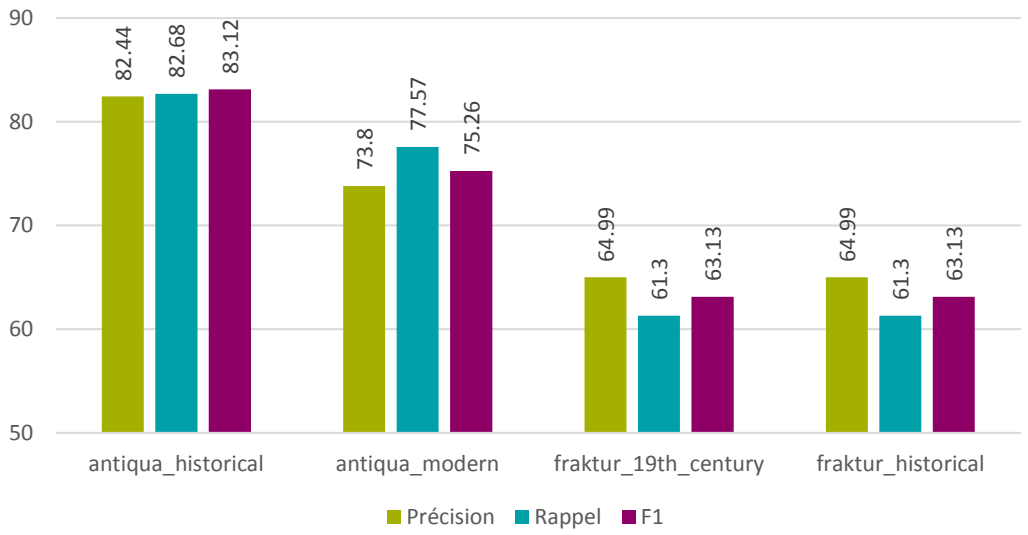
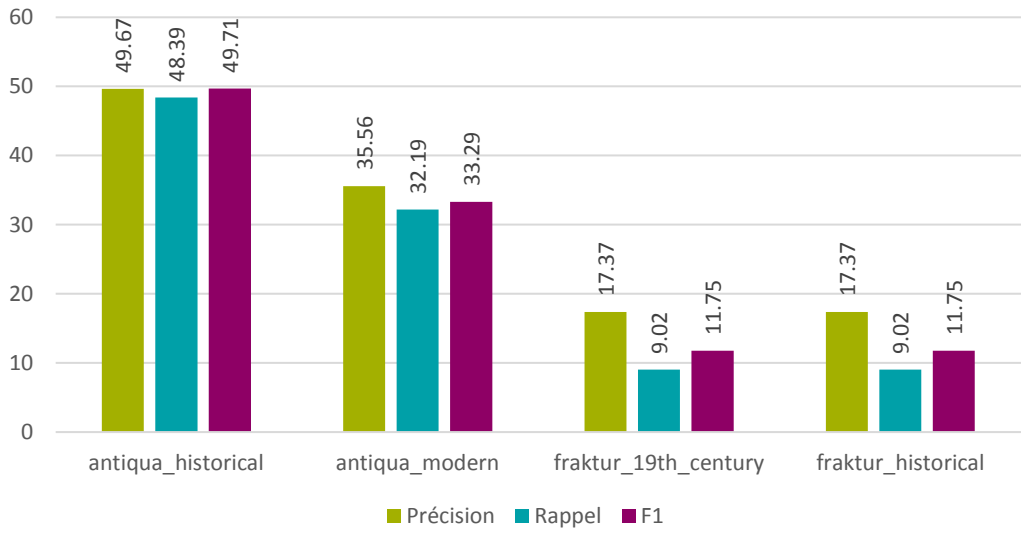


Figure 20 : OCR4all – phase 2 – modèles de typographie – mots



La typographie de la collection de Bry rentre dans la catégorie Antiqua, et il est donc logique que les deux modèles antiqua donnent les meilleurs résultats. Il est en outre assez normal que le modèle antiqua_historical, basé sur des textes de la même époque que la collection de Bry, soit le plus performant. Il était tout de même intéressant de tester les modèles Fraktur, car l'on peut toujours avoir des surprises.

Nous souhaitions ensuite tester différents paramétrages, mais OCR4all propose 26 paramètres différents, tous modulables. Il aurait malheureusement été bien trop chronophages, dans le cadre de ce projet, de tous les tester, mais cela aurait sans doute pu donner des résultats intéressants. Le seul test que nous avons fait en changeant quelques paramètres (seuil de luminosité, précision de la segmentation et choix de l'algorithme décisionnel) a produit de moins bons résultats qu'auparavant, et nous avons donc choisi d'arrêter là l'expérience.

A ce stade, les modèles d'OCR4all sont bien plus performants que ceux des autres logiciels testés.

3.3.4.3 OCR4all – phase 3

Lors de cette dernière phase de test, nous avons utilisé les mêmes méthodes de pré-traitement d'image qu'avec Tesseract. Nous avons encore réussi à augmenter les F_1 de 2.31% au niveau des caractères. Au niveau des mots, nous avons eu une légère baisse de 0.2%.

Figure 21 : OCR4all – phase 3 – modèle antiqua_historical – caractères

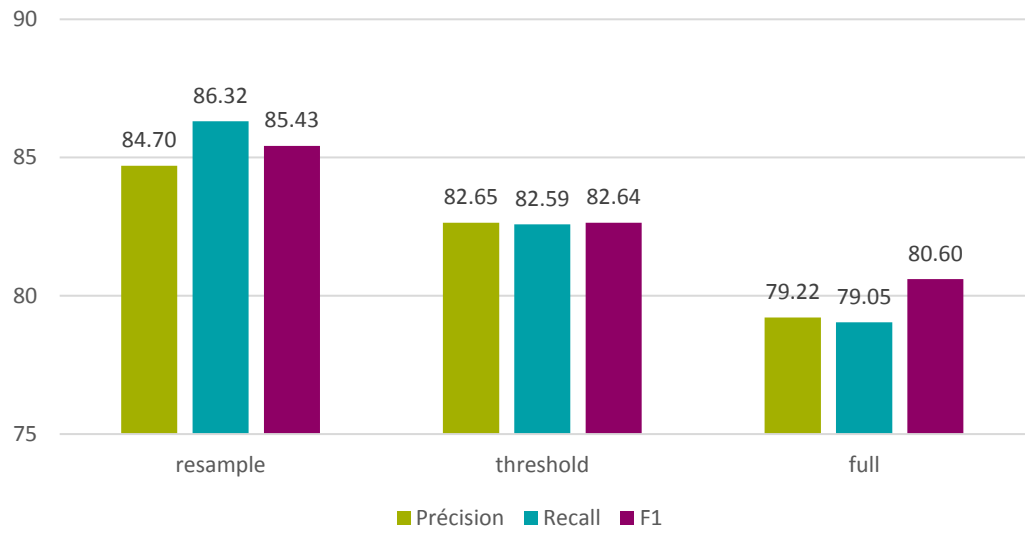
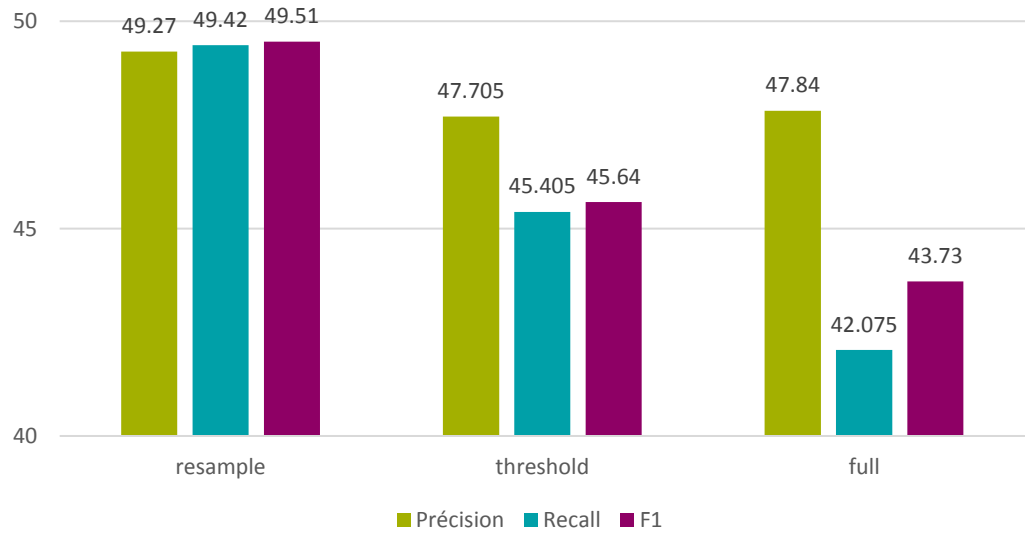


Figure 22 : OCR4all – phase 3 – modèle antiqua_historical – mots

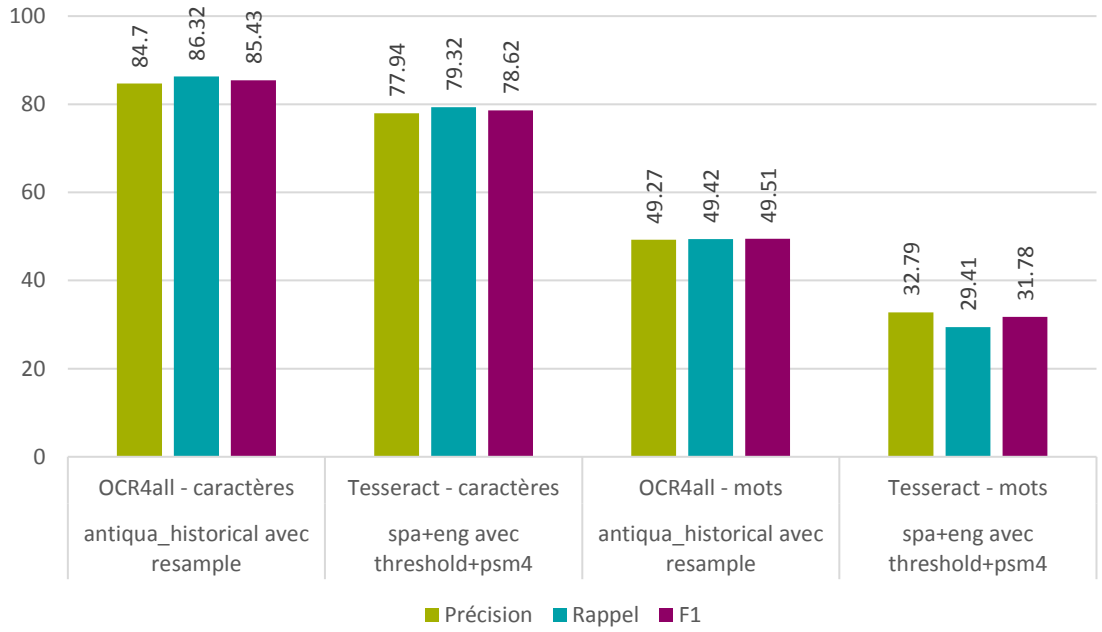


À ce stade, nous pouvons donc compter sur une F_1 de **85.43%** au niveau des caractères et **49.51%** au niveau des mots, avec OCR4all modèle antiqua_historical et la méthode *resample*.

3.4 Sélection

À l'issue de ces tests, nous avons finalement dû décider entre Tesseract et OCR4all. Le graphique ci-dessous récapitule les meilleurs résultats obtenus avec chacun de ces deux logiciels, au niveau des mots et au niveau des caractères.

Figure 23 : comparaison des performances des meilleurs modèles d'OCR4all et Tesseract



Comme on le voit, OCR4all est plus performant que Tesseract, ce qui fait pencher la balance en sa faveur. Malheureusement, il présente le même défaut que Kraken, qui est de tourner en boucle infinie lorsqu'il est confronté à une page blanche. Ce problème pouvant être un obstacle de taille et ralentir considérablement notre travail, surtout au moment de l'océrisation de l'ensemble de la collection de Bry, nous avons préféré porter notre choix sur Tesseract. Nous recommandons cependant de suivre de près l'évolution de la résolution de ce problème, qui pourrait alors faire d'OCR4all modèle antiqua_historical le meilleur choix possible.

4. Océrisation avec Tesseract

La deuxième partie de ce travail consistait à tester différentes méthodes pour améliorer les résultats de l'OCR sélectionné, à savoir Tesseract, en utilisant des techniques de pré-traitement des *inputs*, de post-correction des *outputs*, ou en utilisant des fonctionnalités du logiciel lui-même.

4.1 Méthodes

Nous avons testé plusieurs méthodes :

- Pré-traitement intelligent des images
- Correction brute des outputs
  - Transformation systématique des caractères
  - Suppression de caractères indésirables
  - Suppression des caractères non alphanumériques
- Utilisation d'un corpus latin pour la création d'un dictionnaire
  - Remplacement des mots selon une distance de modification définie
- Utilisation de l'outil de post-correction PoCoTo
- Création d'un modèle Tesseract personnalisé

4.2 Pré-traitement intelligent des images

Comme vu dans Tesseract – phase 3, un pré-traitement intelligent des images peut améliorer les résultats des océrisations.

Pour ce faire, nous avons créé un algorithme de recherche permettant de savoir si la page actuellement traitée est un recto ou un verso. En fonction de cela, un autre algorithme que nous avons développé va trouver la position idéale pour rogner l'image et l'effectuer à l'aide de « ImageMagick », mentionné dans Tesseract – phase 3.

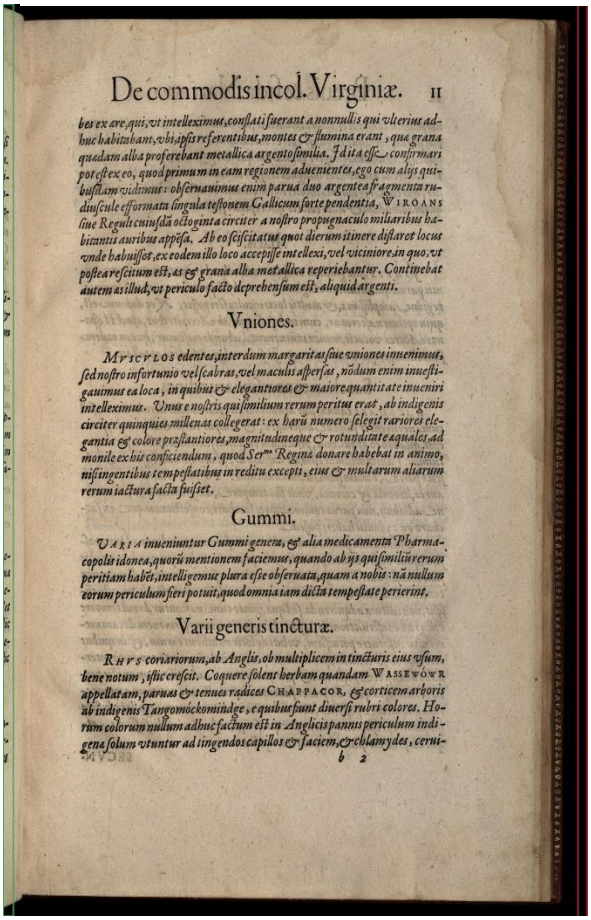
L'algorithme de sélection de l'orientation de la page calcule la somme des couleurs de chaque pixel de la colonne de pixels la plus à droite et la plus à gauche de l'image. La somme la plus petite permet d'indiquer si l'image est un recto ou un verso.

A priori, cet algorithme fonctionne uniquement avec notre jeu de données, car le Bodmer Lab a pour habitude de cadrer ses numérisations en gardant une partie de la page adjacente ainsi qu'un fond noir sur le bord opposé. Ainsi, en déterminant quel côté est "le plus foncé", on peut savoir si la page est un recto ou un verso.

Sur l'image suivante, la partie droite (en rouge) comporte uniquement des pixels noirs. Un pixel noir ayant une valeur d'environ 0 (dépendant de la luminosité de la pièce au moment de la numérisation), la somme sera petite. Inversement, la partie de gauche (en

vert), aura une somme bien plus élevée. On peut donc dire que cette image est un recto car la somme la plus faible est celle de droite.

Figure 24 : détection des zones à rogner selon notre algorithme dans une numérisation extraite de la collection de Bry



De commodis incol. Virginiæ. 11

bes ex ære, qui, vt intelleximus, conflati fuerant a nonnullis qui vlterius adhuc habitabant, vbi, ipsis referentibus, montes & flumina erant, quæ grana quædam alba proferebant metallica argento similia. Id ita esse confirmari potest ex eo, quod primum in eam regionem aduenientes, ego cum alijs quibusdam vidimus: obseruauimus enim parua duo argentea fragmenta rudiuscule efformata singula testonem Gallicum forte pendentia, WIROANS siue Regulus cuiusdã octoginta circiter a nostro propugnaculo militaribus habitantis auribus appensa. Ab eo scisciitatus quot dierum itinere distaret locus vnde habuisset, ex eodem illo loco accepisse intellexi, vel viciniore, in quo, vt postea resciuim est, as & grana alba metallica reperiebantur. Continebat autem as illud, vt periculo facto deprehensum est, aliquid argenti.

Vniones.

MYSCVLOS edentes, interdum margaritas siue vniones inuenimus, sed nostro infortunio vel scabras, vel maculis aspersas, nõdum enim inuestigauimus ea loca, in quibus & elegantiores & maiore quantitate inueniri intelleximus. Vnus e nostris quisimilium rerum peritus erat, ab indigenis circiter quinquies millenas collegerat: ex harũ numero selegit rariores elegantia & colore præstantiores, magnitudineque & rotunditate æquales: ad monile ex his conficiendum, quod Serm. Reginæ donare habebat in animo, nisi ingentibus tempestatibus in rediu exceptis, eius & multarum aliarum rerum iactura facta fuisset.

Gummi.

VARIA inueniuntur Gummi genera, & alia medicamenta Pharmacopolis idonea, quorũ mentionem faciemus, quando ab ijs quisimiliũ rerum peritiam habẽt, intelligemus plura esse obseruata, quam a nobis: nã nullum eorum periculum fieri potuit, quod omnia iam dicta tempestate perierint.

Varii generis tincturæ.

RHVS coriariorum, ab Anglis, ob multiplicem in tincturis eius vsum, bene notum, sic crescit. Coquere solent herbam quandam WASSEWÓWR appellatam, paruas & tenues radices CHAPPACOR, & corticem arboris ab indigenis Tangomockomindge, e quibus fiunt diuersi rubri colores. Horum colorum nullum adhuc factum est in Anglicis pannis periculum indigena solum vtuntur ad tingendos capillos & faciem, & chlamydes, cerui-

b 2

Algorithm 1 Page recto / verso

Input: Chemin de l'image
Ouvrir l'image
Convertir l'image en pixels noirs et blancs
Initialiser right=0, left=0, w=image width et h=image height
for $i = 0$ **to** $h - 1$ **do**
  Initialiser r comme la valeur rouge du pixel[$w - 1$, i]
  Initialiser g comme la valeur verte du pixel[$w - 1$, i]
  Initialiser b comme la valeur bleu du pixel[$w - 1$, i]
  $right+ = r + g + b$

  Définir r comme la valeur rouge du pixel[0, i]
  Définir g comme la valeur verte du pixel[0, i]
  Définir b comme la valeur bleu du pixel[0, i]
  $left+ = r + g + b$
end for
if $right < left$ **then**
  Output: L'image est un recto
else
  Output: L'image est un verso
end if

Lorsque l'on sait si l'image est un recto ou un verso, il faut déterminer quel pourcentage de l'image doit être rogné pour enlever le surplus de la page adjacente. Pour ce faire, nous avons créé un algorithme dont c'est l'objectif. Si la page est recto, l'algorithme va calculer la somme des couleurs de chaque colonne entre la gauche et le centre de l'image. Ces sommes sont stockées dans un tableau. Par la suite, l'algorithme va récupérer l'indice de la valeur la plus faible dans ce tableau. Cet indice signale l'endroit où l'image doit être rognée.

Le processus est presque le même si l'on travaille sur un verso. La seule différence réside dans le calcul des sommes des colonnes de pixels. L'algorithme ne partira pas de la gauche vers le centre mais de la droite vers le centre.

Algorithm 2 Rognage du recto

```
Input: Chemin de l'image d'entrée
Input: Chemin de l'image de sortie
Ouvrir l'image d'entrée
Convertir l'image en pixels noirs et blancs
Initialiser w=image width et h=image height
Initialiser pad=h/100*10
Initialiser values=tableau vide
for i = 0 to w/2 − 1 do
  Initialiser left=0
  for j = pad to h − pad do
    Initialiser r comme la valeur rouge du pixel[i, j]
    Initialiser g comme la valeur verte du pixel[i, j]
    Initialiser b comme la valeur bleu du pixel[i, j]
    left+ = r + g + b
  end for
  Ajouter left dans le tableau values
end for
Initialiser crop comme l'index de la valeur minimum de values
Lancer ImageMagick avec les paramètres récupérés
```

Algorithm 3 Rognage du verso

```
Input: Chemin de l'image d'entrée
Input: Chemin de l'image de sortie
Ouvrir l'image d'entrée
Convertir l'image en pixels noirs et blancs
Initialiser w=image width et h=image height
Initialiser pad=h/100*10
Initialiser values=tableau vide
for i = w − 1 to w/2 with steps −1 do
  Initialiser right=0
  for j = pad to h − pad do
    Initialiser r comme la valeur rouge du pixel[i, j]
    Initialiser g comme la valeur verte du pixel[i, j]
    Initialiser b comme la valeur bleu du pixel[i, j]
    right+ = r + g + b
  end for
  Ajouter right dans le tableau values
end for
Initialiser crop comme l'index de la valeur minimum de values
Lancer ImageMagick avec les paramètres récupérés
```

Figure 25 : image non rognée

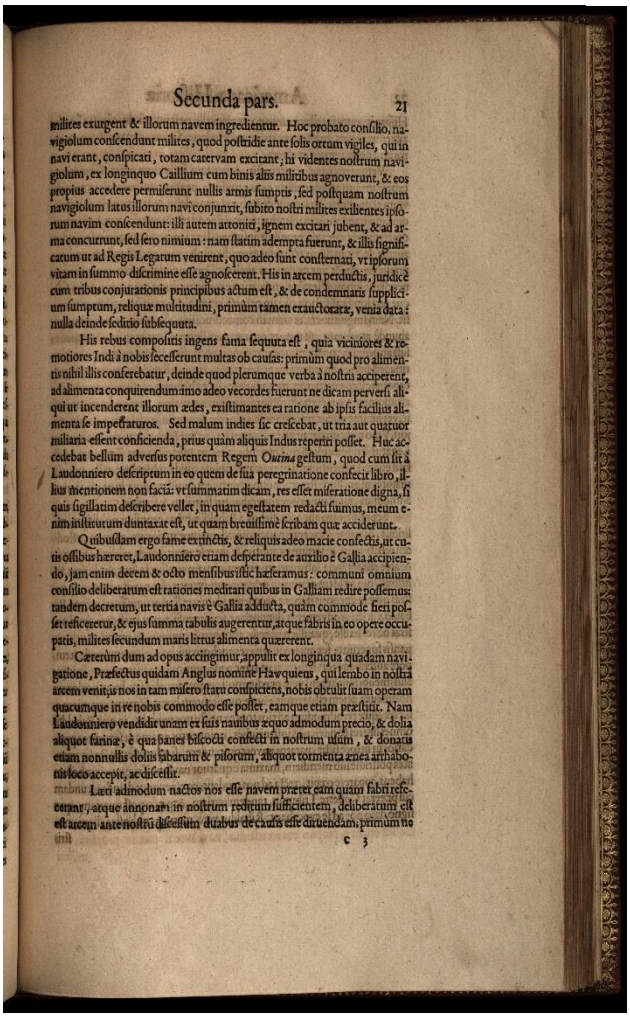
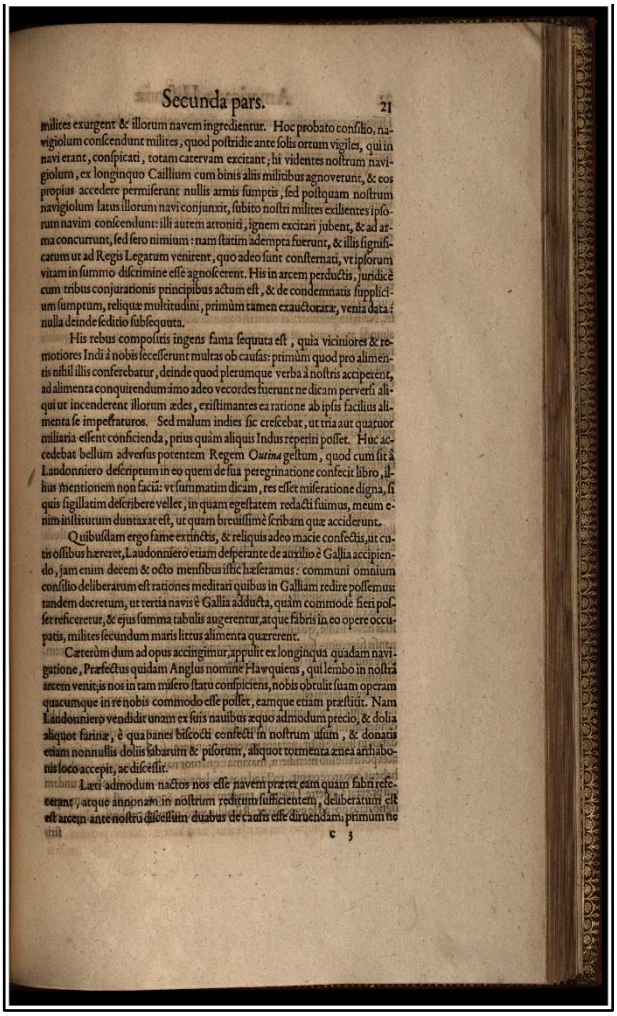


Figure 26 : image rognée avec bordure blanche



Ces deux algorithmes ont également été modifiés de différentes manières pour en créer plusieurs versions. En effet, nous nous sommes aperçus que dans certains cas, le rognage était trop important et qu'une partie du texte était alors perdu. Pour éviter cela, nous avons alors tenté de diviser la valeur du rognage (*crop*) par son quart, son tiers et sa moitié.

L'appel à ImageMagick est le suivant :

Dans le cas d'une image recto :

convert *imagePath* -gravity West -chop *chop*x0 -trim -trim -resample -bordercolor white -border 20x20 *savePath*

Dans le cas d'une image verso :

convert *imagePath* -gravity East -chop *chop*x0 -trim -trim -resample -bordercolor white -border 20x20 *savePath*

Il suffit de remplacer les valeurs italiques soulignées par les valeurs récupérées dans l'algorithme.

Grâce à ces trois algorithmes, nous sommes capables de supprimer la page adjacente sur l'image actuellement traitée. Avec la meilleure version de notre algorithme, nous arrivons à faire passer la F_1 de 78.62% à 79.23% au niveau des caractères et de 31.78% à 32.14% au niveau des mots.

Figure 27 : Tesseract – pré-traitement intelligent – caractères

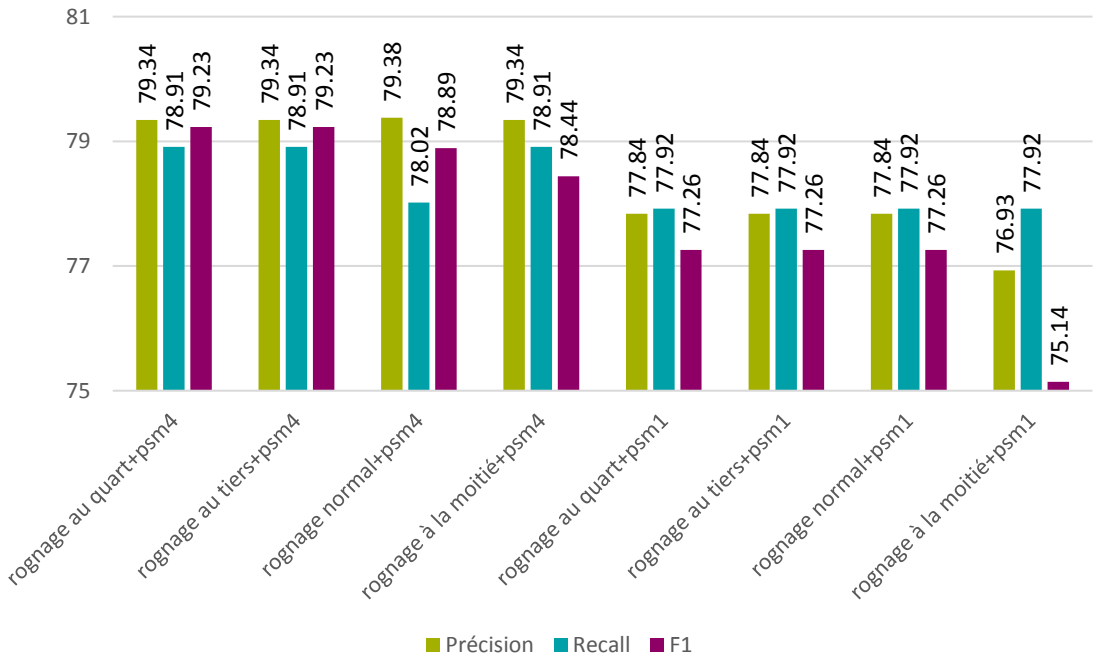
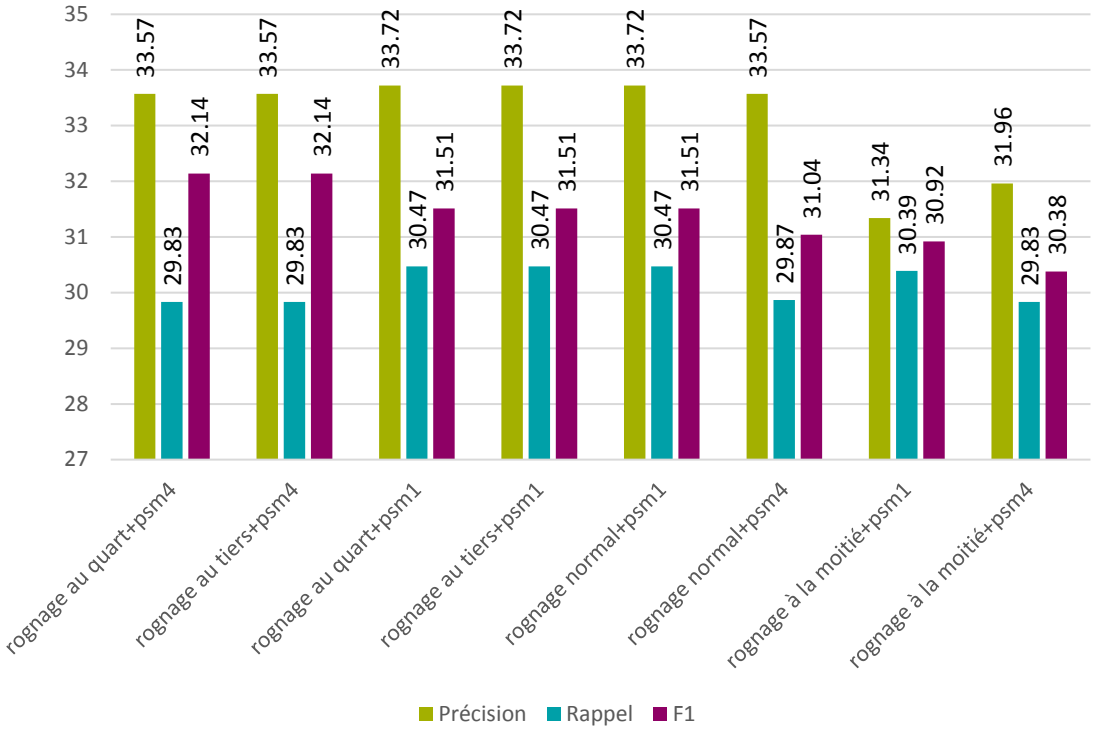


Figure 28 : Tesseract – pré-traitement intelligent – mots



4.3 Correction brute des *outputs*

Après avoir appliqué les algorithmes de pré-traitement et océrisé l'ensemble des images rognées dans Tesseract, il est possible d'améliorer encore les résultats en effectuant une post-correction sur les *outputs*.

Pour ce faire, nous avons testé plusieurs algorithmes :

1. Suppression de tous les caractères différents d'un caractère d'espacement (vertical ou horizontal), d'une lettre (minuscule ou majuscule) ou d'un chiffre
   - Par la suite, remplacement de tous les v par des u et les j par des i
2. Uniquement de tous les v par des u et les j par des i
3. Suppression de tous les caractères différents d'une lettre (minuscule ou majuscule), d'un chiffre, d'un point, d'une virgule, d'un espace ou d'un retour à la ligne
   - Par la suite, remplacement de tous les v par des u et les j par des i
4. Suppression de tous les caractères différents d'une lettre (minuscule ou majuscule), d'un chiffre, d'un double point, d'un tiret, d'un espace ou d'un retour à la ligne
   - Par la suite, remplacement de tous les v par des u et les j par des i

5. Suppression de tous les caractères différents d'une lettre (minuscule ou majuscule), d'un double point, d'un tiret, d'un espace ou d'un retour à la ligne

- Par la suite, remplacement de tous les v par des u et les j par des i

Les résultats montrent qu'un simple remplacement des lettres v et j améliore les résultats. En effet, nous passons d'une F_1 de 79.23% à une F_1 de 80.06% au niveau des caractères, et de 32.14% à 34.58% au niveau des mots.

Figure 29 : Tesseract – correction brute – caractères

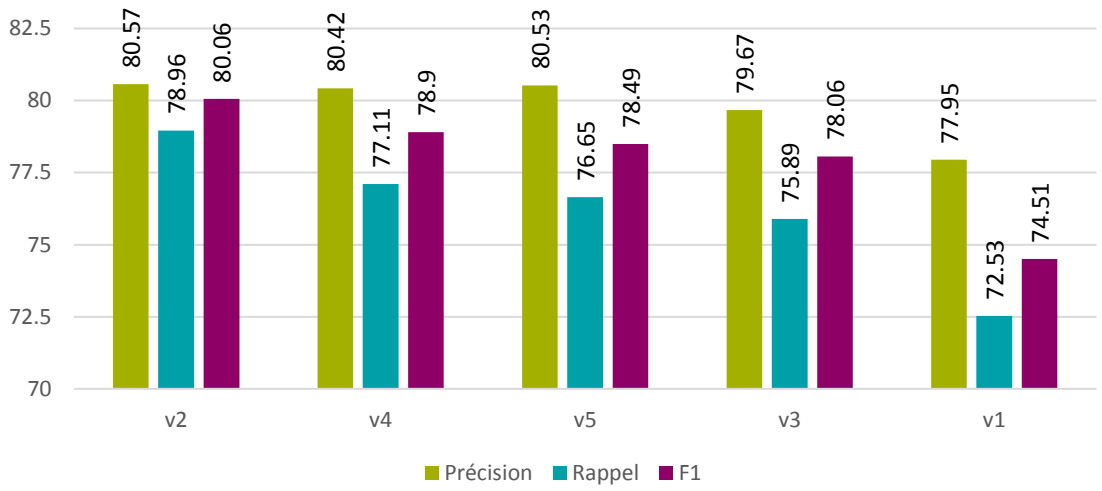
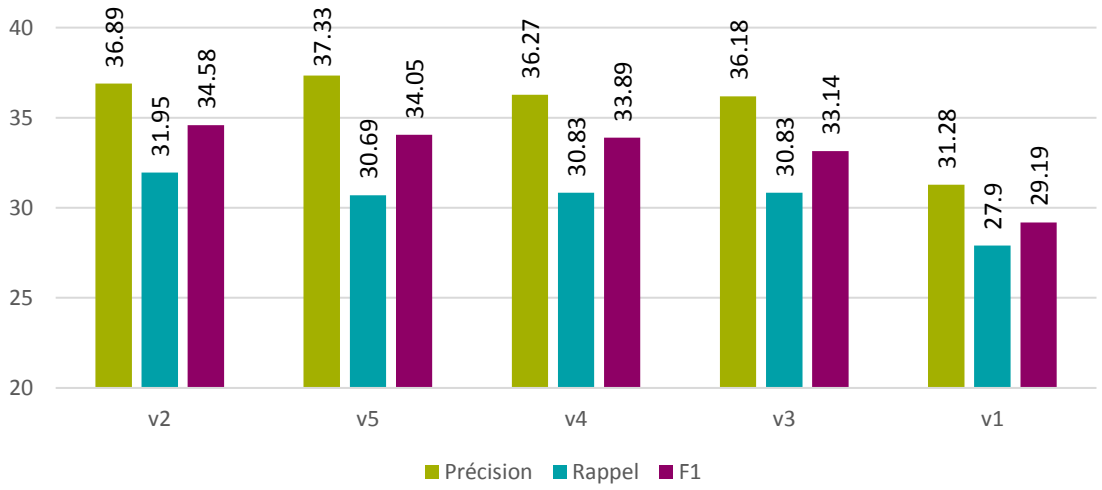


Figure 30 : Tesseract – correction brute – mots



4.4 Utilisation d'un corpus latin pour la création d'un dictionnaire

Une autre méthode de post-correction envisageable était la modification directe des mots sur la base d'un corpus de textes latins. Nous avons utilisé la librairie « symspellpy », disponible sur pypi.org/project/symspellpy, afin de créer le dictionnaire et de calculer la distance d'édition des mots. Pour le corpus, nous avons utilisé celui de Latinocr.org, mentionné dans notre État de l'art, qui se trouve sur ryanfb.github.io/latinocr/resources.html.

Nous avons ensuite créé l'algorithme suivant :

Algorithm 4 Remplacement par dictionnaire

```
Input: Text a corriger
Instancier Symspellpy
Télécharger ou créer le dictionnaire latin en se basant sur un corpus de textes
latins
Initaliser lines=tableau vide
Initialiser text_corrected=""
Couper notre text à chaque retour de ligne et mettre chaque valeur dans le
tableau lines
for i = 0 to longueur de lines - 1 do
  Initialiser words=tableau vide
  Couper lines[i] à chaque espace et mettre chaque valeur dans le tableau
  words
  while j < longeur de words - 2 do
    Initialiser s1 comme étant la suggestion la plus proche de words[j] en
    utilisant symspellpy
    Initialiser s2 comme étant la suggestion la plus proche de words[j] +
    words[j + 1] en utilisant symspellpy
    if distance de s1 < distance de s2 then
      Définir text_corrected=s1 + " "
      Définir j = j + 1
    else
      Définir text_corrected=s2 + " "
      Définir j = j + 2
    end if
  end while
Définir text_corrected=text_corrected + "\n"
end for
Output: Le texte corrigé "text_corrected"
```

Cet algorithme utilise la librairie « symspellpy » pour comparer chaque mot avec ceux du dictionnaire. Pour ce faire, il est nécessaire de définir une distance de recherche maximale (2, dans notre cas). Cette distance limite les résultats aux mots qui ont une distance d'édition de maximum deux caractères.

Nous avons également remarqué que Tesseract trouve parfois des espaces au milieu des mots. Afin de pouvoir les corriger automatiquement avec notre algorithme, nous

avons décidé de comparer chaque mot et celui qui le suit avec le dictionnaire. De cette manière, si un mot a été coupé en deux, nous pouvons le traiter.

Nous avons testé plusieurs versions de cet algorithme :

1. Distance de suggestion du mot < distance de suggestion du mot et de celui qui le suit
2. Distance de suggestion du mot <= distance de suggestion du mot et de celui qui le suit
3. Distance de suggestion du mot et de celui qui le suit < distance de suggestion du mot
4. Distance de suggestion du mot et de celui qui le suit <= distance de suggestion du mot

Aucune de ces versions n'a pu améliorer nos résultats. Au contraire, ils baissent d'environ 5%. Cela est dû au fait que chaque mot va essayer d'être corrigé par rapport à un mot du corpus – même s'il est correct, pour autant que la distance d'édition soit inférieure ou égale à 2. De ce fait, la précision, le rappel et la F_1, que ce soit au niveau des caractères ou des mots, baissent.

Figure 31 : Tesseract – corrections par dictionnaire – caractères

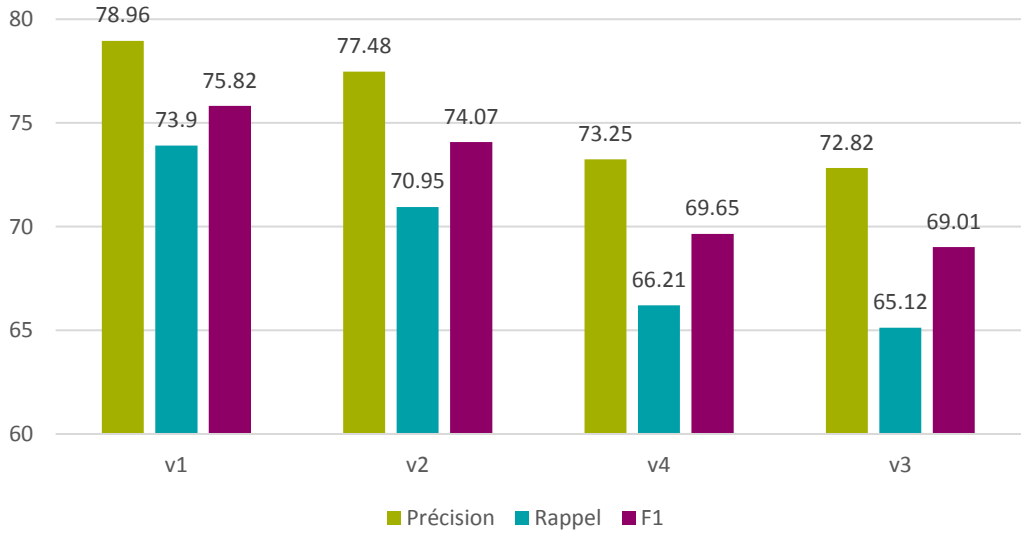
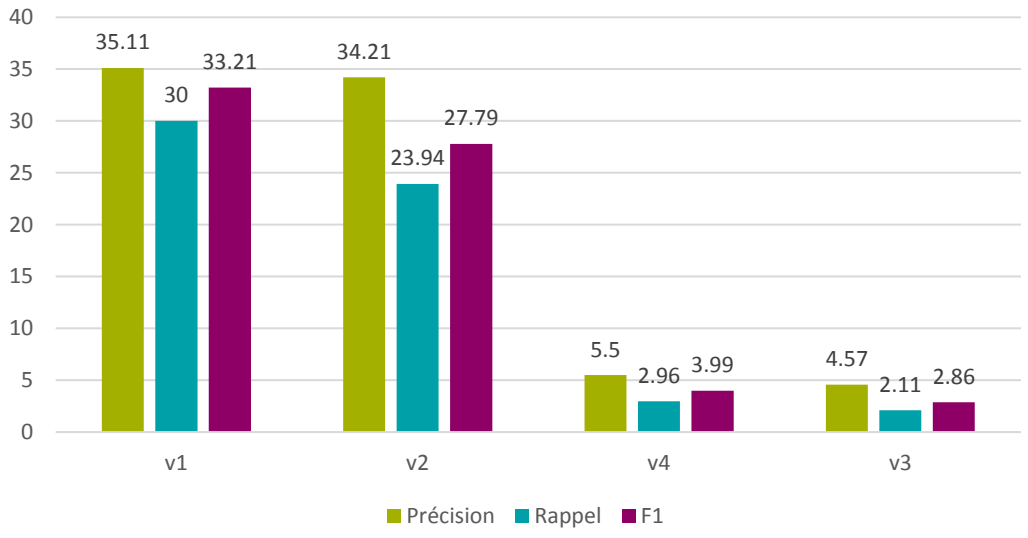


Figure 32 : Tesseract – corrections par dictionnaire – mots



Ce problème peut s'expliquer entre autres par la complexité du latin, qui est une langue à cas. Dans la plupart des dictionnaires latins, un nom va être présenté sous la forme *consul, -is, m.*, ce qui veut dire que c'est un mot masculin de la 3ème déclinaison, qui prend donc *-is* comme terminaison au génitif. Ce mot pourrait cependant apparaître sous des formes comme *consules* ou *consulibus*, mais le dictionnaire ne contient pas ces formes, évidentes pour un latiniste, mais non pour un ordinateur.

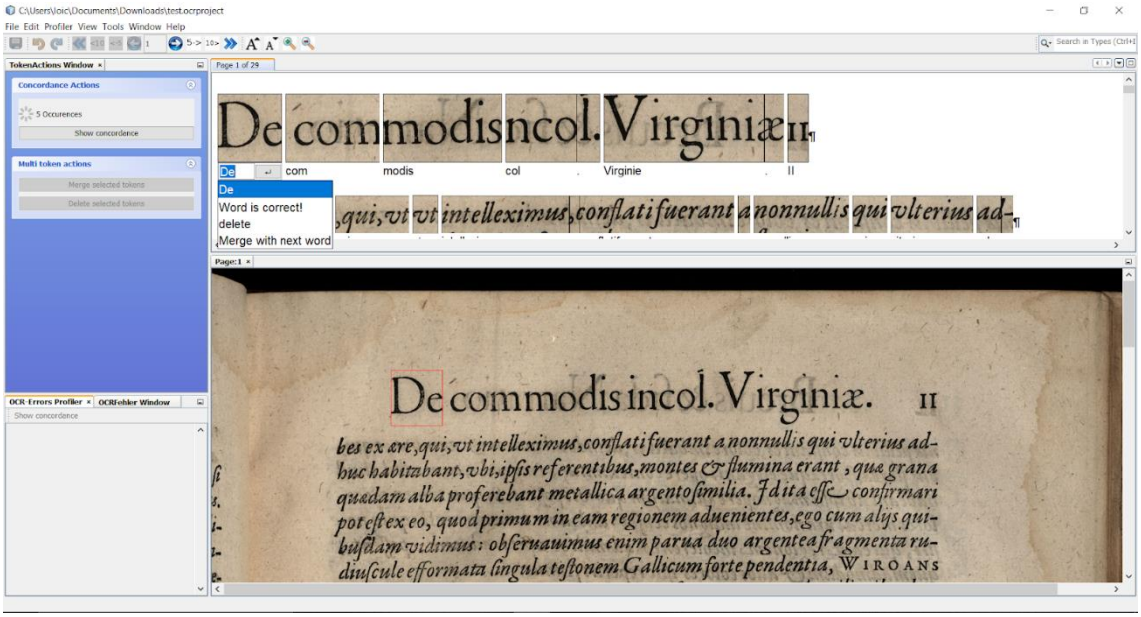
Il aurait cependant été intéressant de pousser plus loin l'expérience des dictionnaires, méthode de post-correction reconnue dans le domaine de l'océrisation, mais, pour des raisons de faisabilité, nous avons dû nous en tenir là.

4.5 Utilisation de l'outil de post-correction PoCoTo

Une autre méthode de post-correction que nous souhaitions tester consistait à utiliser le logiciel PoCoTo. Il s'agit d'un logiciel de post-correction développé dans le cadre du projet IMPACT et permettant de corriger les erreurs des logiciels OCR (Vobl et al. 2014). Nous avons testé ce logiciel et nous sommes vite aperçus de ces avantages et limites.

PoCoTo prend en *input* les images a océriser ainsi que leur océrisation au format HOCR. Ce format enregistre l'océrisation de chaque caractère (comme avec le format texte) mais également la position de la portion d'image qui lui a fait découvrir ce caractère. Il est donc possible par la suite, grâce à PoCoTo, de vérifier manuellement si l'*output* correspond au *ground truth* et de la corriger au besoin. Ceci est cependant long, car l'on corrige chaque mot un à un.

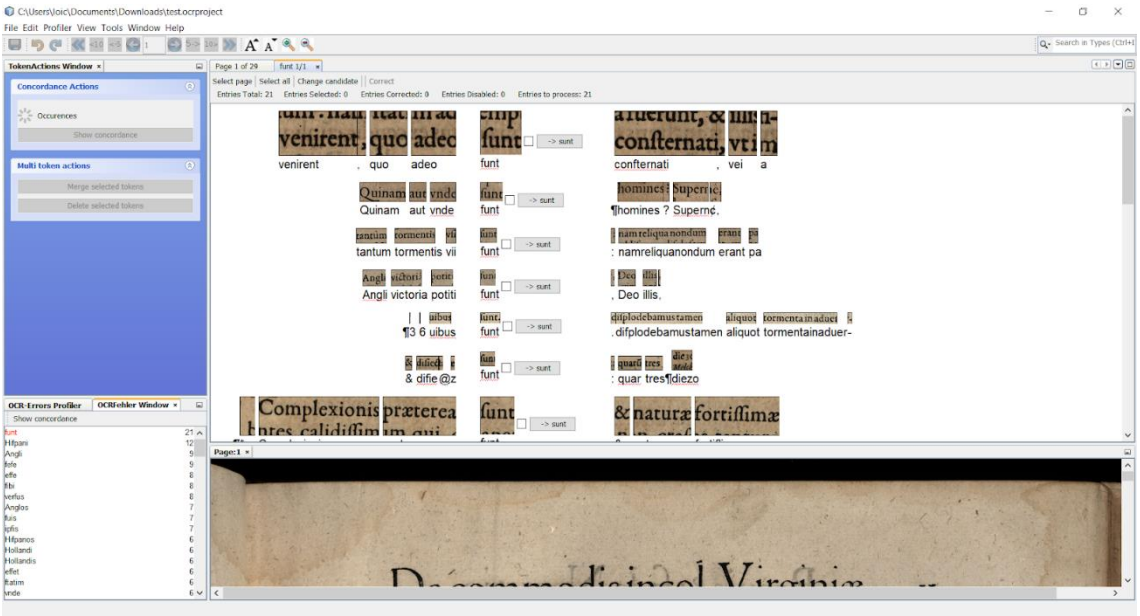
Figure 33 : capture d'écran du logiciel PoCoTo en cours d'utilisation



Une seconde option de PoCoTo permet de télécharger des *profilers*. Actuellement, il est possible de télécharger des *profilers* en latin, en grec ou en allemand pré-entraînés. Il est également possible de créer son propre *profiler*.

Ces derniers stockent les erreurs courantes des OCR pour un langage en particulier, afin d'effectuer une correction « semi-automatique » des *outputs*. Le *profiler* va détecter si un mot souvent reconnu incorrectement par les OCR se trouve dans l'un des *outputs* et propose une correction. Néanmoins, tout se fait depuis une interface graphique et une intervention humaine est alors obligatoire.

Figure 34 : capture d'écran du logiciel PoCoTo en cours d'utilisation avec le système de *profiler* latin



Cette obligation d'avoir une intervention humaine est chronophage et, au vu du temps qu'il nous restait pour ce projet, nous avons décidé de ne pas intégrer cet outil dans notre post-correction.

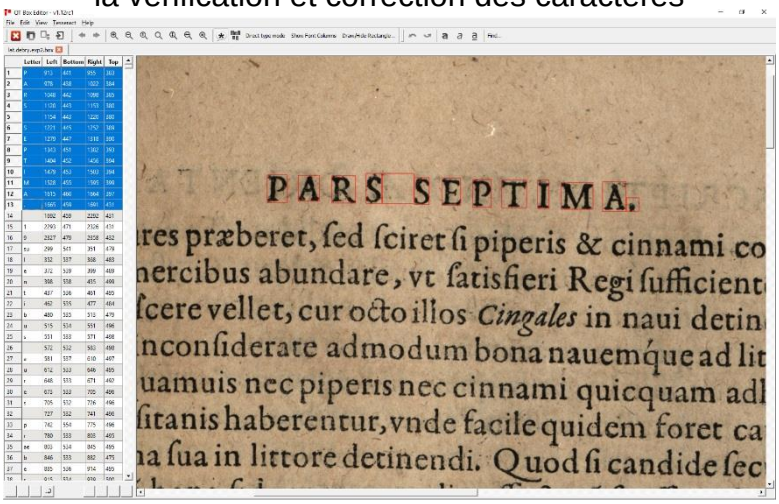
4.6 Création d'un modèle Tesseract personnalisé

La dernière méthode d'optimisation des résultats que nous avons testée est la création de notre propre modèle Tesseract. Cette fonctionnalité est rendue possible par l'outil *open source* multi-plateforme, QT Box Editor, disponible sur github.com/zdenop/qt-box-editor. Cet outil nous offre la possibilité de corriger manuellement la segmentation des caractères sur la numérisation ainsi que chaque caractère identifié, afin de pouvoir créer notre propre modèle basé sur les typographies de la collection de Bry.

Nous avons essayé cette méthode en dernier car la création d'un modèle personnalisé prend un temps considérable mais n'assure pas pour autant d'améliorer les résultats. En outre, il y a de fortes chances que notre modèle ne soit pas réutilisable pour d'autres collections, étant donné que nous apprenons au logiciel à reconnaitre les typographies spécifiques de cette collection.

Tesseract préconise d'avoir au minimum trois fois chaque caractère dans notre jeu d'entraînement, et nous avons alors sélectionné trois images contenant la plupart des caractères pour créer notre jeu de données. Une de ces images comporte une gravure afin que Tesseract apprenne également à ne pas y reconnaître de texte. Finalement, le travail fourni sur ces trois numérisations correspond à une vérification et correction manuelle d'environ 6'200 caractères.

Figure 35 : capture d'écran de QT Box Editor pendant la vérification et correction des caractères



Notre ébauche de modèle nous a permis d'obtenir des résultats relativement positifs, au vu du peu de données ayant servi à sa création, mais c'est bien sûr insuffisant par rapport à nos autres tests. L'idée reste cependant intéressante, et, si le temps permet de traiter quelques images de plus, les résultats pourraient peut-être dépasser ceux des modèles testés auparavant.

Figure 36 : Tesseract – modèle personnalisé – caractères

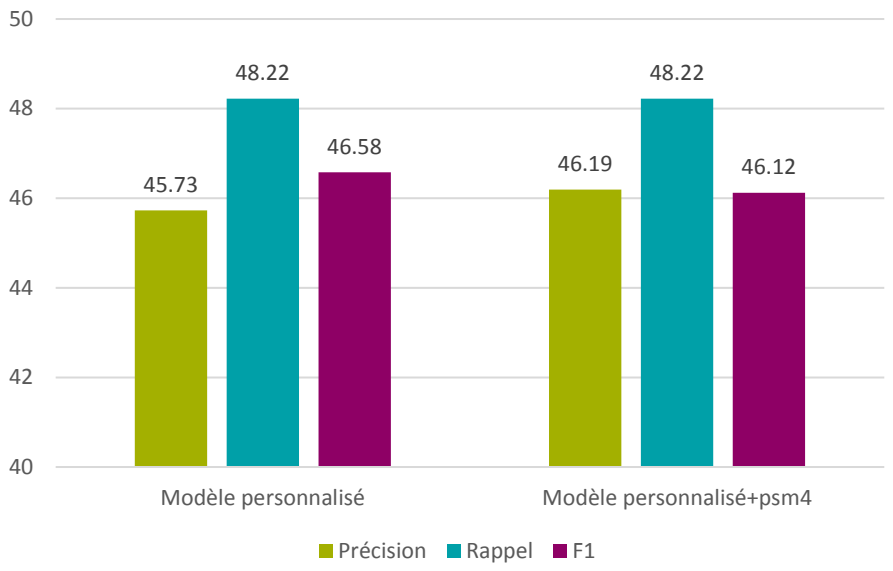
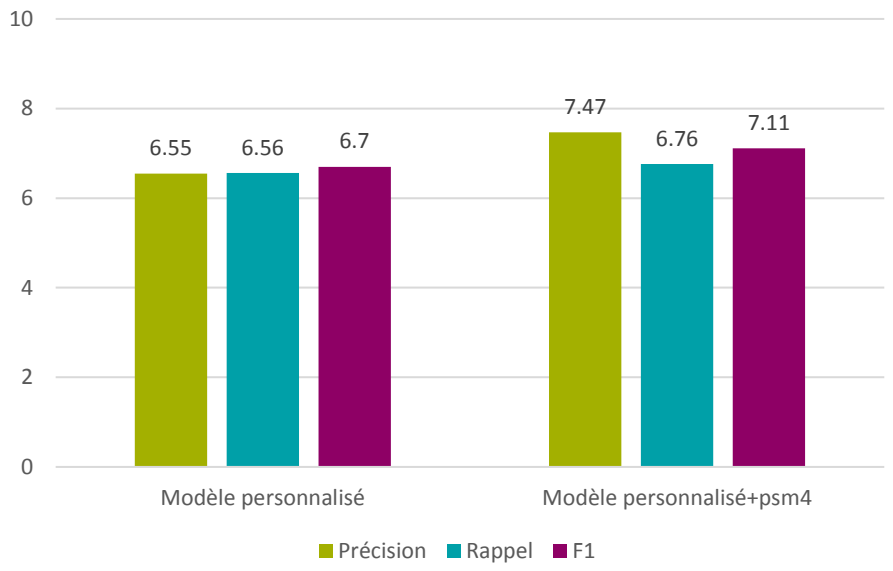


Figure 37 : Tesseract – modèle personnalisé – mots



4.7 Résultats finaux

À la suite de tous ces tests, nous parvenons à une F_1 maximale de **80.06%** au niveau des caractères, et de **34.58%** au niveau des mots. L'objectif de 95% n'est pas atteint, mais nous avons pu tout de même nous en rapprocher.

5. Intégration des transcriptions

La troisième partie de ce projet consiste à proposer des solutions pour intégrer les transcriptions sur le site web du Bodmer Lab, afin de rendre leur consultation aisée et ergonomique. Pour ce faire nous avons commencé par nous renseigner sur les technologies utilisées actuellement par le Bodmer Lab.

5.1 Outils actuels

Les numérisations des différentes constellations sont disponibles sur le site grâce à la plateforme de visionnage d'image Mirador (Mirador 2020) et aux spécifications IIIF (IIIF 2019) qui la sous-tendent. Ceci rend ces images faciles à visionner et à utiliser grâce à l'interopérabilité qu'offre IIIF.

Certaines constellations du Bodmer Lab proposent, en plus du simple visionnage, un accès aux métadonnées des documents grâce à un onglet dédié. Cependant aucune constellation ne présente de transcription, à l'exception des papyrus de Ménandre, où une forme d'interaction avec le texte est possible. Ceci reste très spécifique au type de document en question, très différent de la collection de Bry, et l'idée n'est pas transférable en l'état.

Nous avons alors cherché des exemples d'utilisation de IIIF et de ses visionneuses, comme Mirador, par des institutions similaires et avec un même objectif – rendre des numérisations accessibles pour la recherche plein texte.

Pour ce faire, nous avons consulté la liste des *participating institutions*, disponible sur le site de IIIF (IIIF 2020), et exploré les sites web de celles qui nous semblaient proches du Bodmer Lab en termes de documents et de public cible. Les quelques exemples ci-dessous présentent des utilisations de IIIF qui ont retenu notre attention.

5.2 Exemples existants

5.2.1 E-Rara et Gallica

Près de chez nous, la plateforme *E-Rara*, qui présente « les imprimés numérisés des bibliothèques suisses » (e-rara 2020), et la plateforme *Gallica*, la bibliothèque numérique de la Bibliothèque nationale de France (Bibliothèque nationale de France 2020), utilisent toutes les deux IIIF et ses visionneuses pour présenter leurs documents. Ces dernières proposent un onglet donnant accès aux métadonnées des documents, mais aucune ne propose de transcription.

5.2.2 Europeana

La bibliothèque numérique de l'Union Européenne lancée en 2008, *Europeana* (Europeana 2020), présente également ses journaux numérisés avec une visionneuse IIIF. Certains des documents proposent une recherche plein texte dans une transcription située à droite de l'image. Il est également possible de sélectionner un paragraphe ou un mot dans l'image, et son équivalent apparaît alors dans la transcription.

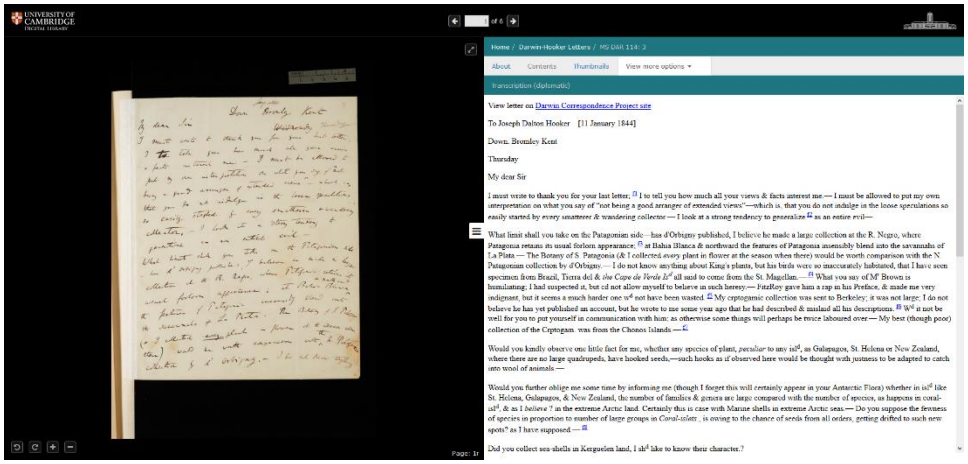
Figure 38 : capture d'écran du site d'Europeana

Ce type de fonctionnalité est possible grâce à des plug-ins de la visionneuse, gratuits et *open source* tout comme elle.

5.2.3 Cambridge Digital Library

La bibliothèque de l'Université de Cambridge propose une bibliothèque numérique, dont les numérisations sont visualisables avec des visionneuses IIIF (University of Cambridge 2015). Un onglet sur le côté donne accès aux métadonnées, mais aussi aux transcriptions quand celles-ci sont disponibles. Une simple recherche type Ctrl+F y est possible.

Figure 39 : capture d'écran du site de la Cambridge Digital Library



Cet onglet propose en outre de choisir entre une transcription normalisée ou diplomatique. Cette possibilité est offerte par un plug-in de la plateforme de visualisation IIIF, et donc accessible gratuitement.

5.2.4 Les Manuscrits de Stendhal

Notre dernier exemple, celui du projet de numérisation et de transcription des manuscrits de Stendhal à Grenoble (Lebarbé et al. 2014) se distingue des cas précédents par le fait que IIIF n'y est pas utilisé. Le visuel est peu esthétique et la recherche y est peu ergonomique, mais il est cependant intéressant de noter que cette plateforme offre également différentes formes de transcriptions (linéaires, pseudo-diplomatique etc.). Bien qu'il s'agisse d'une problématique plus directement liée aux manuscrits qu'aux imprimés, il est tout de même intéressant d'envisager la possibilité de proposer plusieurs transcriptions pour servir différents usages.

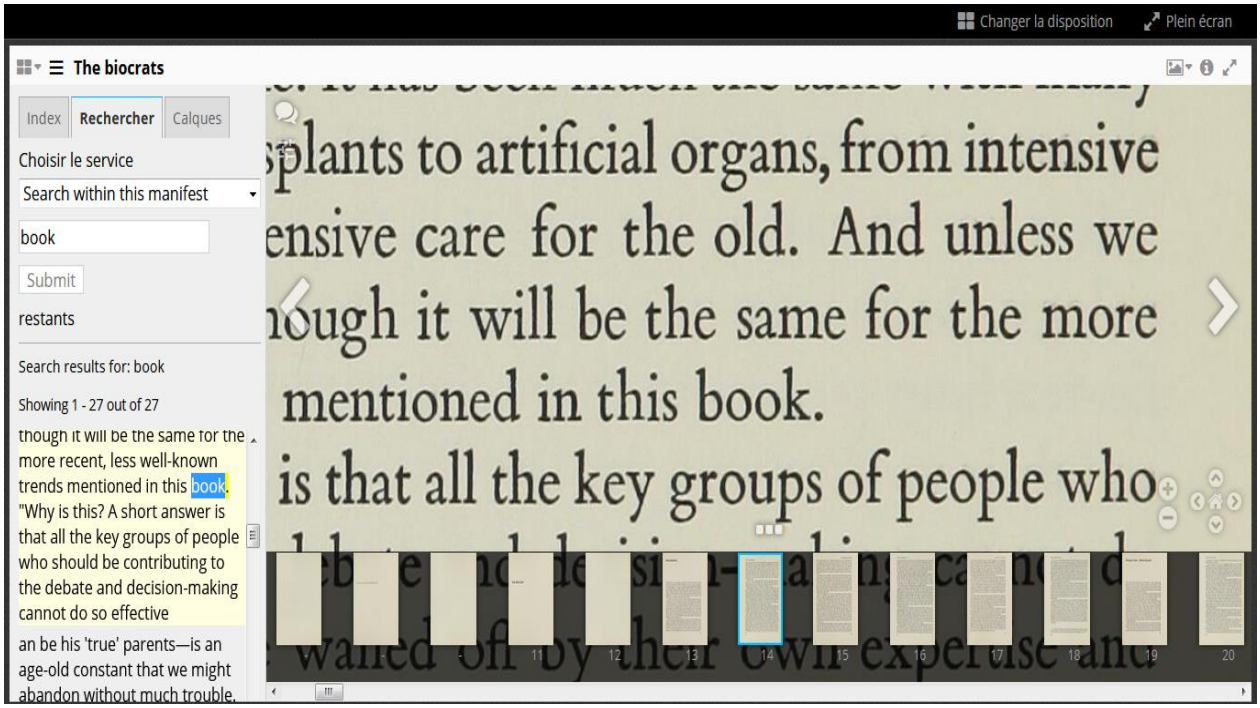
5.3 Recommandations

À la suite de nos recherches, notre premier constat a été que l'utilisation de IIIF et de ses plateformes de visualisations que fait déjà le Bodmer Lab est un excellent point de départ, et qu'il serait futile de proposer une solution différente sans exploiter tout le potentiel de celle déjà mise en place. Nous suggérons donc d'utiliser les fonctionnalités et plug-ins proposés par les plateformes de visionnage IIIF.

5.3.1 Mirador

La plateforme Mirador, déjà utilisée par le Bodmer Lab, possède une fonctionnalité de recherche qu'il suffit de configurer dans les options de *layout*. Il s'agit de la fonctionnalité « *searchTabAvailable* ». Celle-ci permet d'intégrer une traduction à l'onglet de métadonnées et d'y rechercher du texte. En cliquant sur un terme dans la transcription, il est ensuite possible d'accéder directement à sa position dans le document numérisé. Cette option ne coûte rien ni en temps ni en argent, et l'équipe technique du Bodmer Lab devrait pouvoir l'envisager sans difficulté.

Figure 40 : capture d'écran d'un document ouvert avec Mirador, où la recherche plein texte a été activée



5.3.2 Universal Viewer

Il serait également envisageable d'utiliser une autre visionneuse IIIF, comme Universal Viewer (Universal Viewer 2019), qui propose différents plug-ins de recherche en ligne efficients. Cela ne coûterait rien non plus, mais pourrait prendre plus de temps, car il s'agit de mettre en place une autre plateforme de visionnage en plus de Mirador, les deux n'étant pas incompatibles. Il s'agit néanmoins d'une opération relativement simple et rapide.

5.3.3 Content Search API

IIIF propose également une API dédiée à la recherche plein texte (Ronallo 2018) et compatible avec Mirador et Universal Viewer. Cette option, encore une fois gratuite et peu demandeuse en termes de temps, offre un choix plus large d'options, telles que l'auto-complétion des termes de recherche, entre autres.

Ces trois suggestions ne sont pas incompatibles les unes avec les autres, et sont tout à fait à la portée du Bodmer Lab. Nous recommandons de bien clarifier quelles fonctionnalités de recherche plein texte seraient désirées, et de choisir ensuite la ou les options les plus adéquates – qui devraient dans tous les cas être relativement simples et rapides à implémenter.

6. Conclusion et perspectives futures

Ce projet d'océrisation nous a menés à nous aventurer sur des terrains peu ou pas connus de nous – les logiciels OCR, les technologies et méthodes de travail qui leur sont liées, le traitement des imprimés anciens, la complexité de la langue latine etc. Si certains d'entre nous étaient plus à l'aise avec un sujet ou un autre, nous avons tous été amenés à découvrir ce domaine complexe des sciences de l'information et des humanités numériques qu'est l'océrisation. Finalement, même si nous étions novices en la matière, nous sommes tout de même parvenus à des résultats tout à fait honorables et intéressants.

Après avoir testé quatre logiciels d'océrisation, dont deux nous ont très vite posé des problèmes techniques (Kraken et Calamari), nous avons retenu deux logiciels offrant de bonnes performances. Tesseract, lors de notre meilleur test, atteint une F_1 de **78.62%** au niveau des caractères et de **31.78%** au niveau des mots (voir Tesseract – phase 3). OCR4all, de son côté, atteint une F_1 de **85.43%** au niveau des caractères et **49.51%** au niveau des mots (voir OCR4all – phase 3), mais présente un problème technique qui le met malheureusement hors course.

En utilisant différentes méthodes de pré-traitement et de post-correction, nous avons réussi à faire monter les résultats de Tesseract à une F_1 de **80.06%** au niveau des caractères et de **34.58%** au niveau des mots (voir Résultats finaux). Le chemin est encore long jusqu'au 95%, mais nous espérons avoir ouvert la voie à de futurs essais.

Ce projet était limité dans le temps, et il a été frappant de découvrir la durée nécessaire à ce type de travail, chaque paramètre modifié nécessitant une nouvelle itération et un nouveau temps de calcul. Il est de ce fait compréhensible qu'une technologie aussi ancienne soit toujours en développement, du fait de sa complexité et de l'immense variété des données qu'elle traite.

Au terme de ce projet, nous allons livrer au Bodmer Lab l'océrisation de la collection complète avec le meilleur modèle que nous avons pu obtenir. Si ce projet devait continuer dans un autre contexte, ou si d'autres collections devait être océrisées à l'avenir, nous pourrions nous pencher plus sérieusement sur les possibilités qu'offrent la création d'un modèle personnalisé.

En effet, selon les figures de la partie Création d'un modèle Tesseract personnalisé, les performances de notre modèle ne sont pas désastreuses, et mériteraient une amélioration en utilisant plus de données d'entraînement. Nous conseillerons en outre de suivre l'évolution du problème technique posé par OCR4all car, s'il est réglé, ce logiciel et son modèle antiqua_historical pourraient alors devenir le meilleur choix possible.

Bibliographie

AFROGE, Shyla, AHMED, Boshir et MAHMUD, Firoz, 2016. Optical character recognition using back propagation neural network. In : *2nd International Conference on Electrical, Computer Telecommunication Engineering (ICECTE), Rajshahi, 8-10 décembre 2016* [en ligne]. Décembre 2016. pp. 1–4. [Consulté le 28 août 2019]. Disponible à l'adresse : https://ieeexplore.ieee.org/document/7879615

ALGHAMDI, Mansoor et TEAHAN, William, 2017. Experimental evaluation of Arabic OCR systems. *PSU Research Review* [en ligne]. 28 novembre 2017. Vol. 1, no. 3, pp. 229–241. [Consulté le 28 août 2019]. Disponible à l'adresse : http://www.emeraldinsight.com/doi/10.1108/PRR-05-2017-0026

AMIN SHAYEGAN, Mohammad et AGHABOZORGI, Saeed, 2014. A new method for Arabic/Farsi numeral data set size reduction via modified frequency diagram matching. *Kybernetes* [en ligne]. 29 avril 2014. Vol. 43, n°5, pp. 817–834. [Consulté le 28 août 2019]. Disponible à l'adresse : http://www.emeraldinsight.com/doi/10.1108/K-10-2013-0226

ANUGRAH, Rio et BINTORO, Ketut Bayu Yogha, 2017. Latin letters recognition using optical character recognition to convert printed media into digital format. *Jurnal Elektronika Dan Telekomunikasi* [en ligne]. Décembre 2017. Vol. 17, n°2, pp. 56–62. [Consulté le 28 août 2019]. Disponible à l'adresse : https://www.jurnalet.com/jet/article/view/163

ARVINDPDMN [pseudonyme], 2019. Levenshtein distance. *Developedia* [en ligne]. 3 septembre 2019. Mis à jour le 4 septembre 2019. [Consulté le 11 novembre 2019]. Disponible à l'adresse : https://devopedia.org/levenshtein-distance

BALK, Hildelies et PLOEGER, Lieke, 2009. IMPACT : working together to address the challenges involving mass digitization of historical printed text. *OCLC Systems & Services: International digital library perspectives* [en ligne]. 30 octobre 2009. Vol. 25, n°4, pp. 233–248. [Consulté le 28 août 2019]. Disponible à l'adresse : https://www.emeraldinsight.com/doi/full/10.1108/10650750911001824

BAO, Ping et ZHU, Suoling, 2014. System design for location name recognition in ancient local chronicles. *Library Hi Tech* [en ligne]. 10 juin 2014. Vol. 32, n°2, pp. 276–284. [Consulté le 28 août 2019]. Disponible à l'adresse : http://www.emeraldinsight.com/doi/10.1108/LHT-07-2013-0101

BEINERT, Wolfgang 2018. Antiqua. *Typolexicon* [en ligne]. 1er avril 2018. [Consulté le 6 janvier 2020]. Disponible à l'adresse : https://www.typolexikon.de/antiqua/

BEINERT, Wolfgang 2019. Fraktur. *Typolexicon* [en ligne]. 1er août 2019. [Consulté le 6 janvier 2020]. Disponible à l'adresse : https://www.typolexikon.de/fraktur-schrift/

BIBLIOTHÈQUE NATIONALE DE FRANCE, 2020. *Gallica* [en ligne]. 11 janvier 2020. [Consulté le 11 janvier 2020]. Disponible à l'adresse : https://gallica.bnf.fr

BLANKE, Tobias, BRYANT, Michael et HEDGES, Mark, 2012. Open source optical character recognition for historical research. *Journal of Documentation* [en ligne]. Août 2012. Vol. 68, n°5, pp. 659–683. [Consulté le 28 août 2019]. Disponible à l'adresse : https://www.emeraldinsight.com/doi/full/10.1108/00220411211256021

BODMER LAB, 2019. *Bodmer Lab* [en ligne]. 2019. Mis à jour le 9 janvier 2020. [Consulté le 9 janvier 2020]. Disponible à l'adresse : https://bodmerlab.unige.ch/fr

BOWERS, Steven K., 2018. Information Technology and Libraries at 50 : The 1990s in *Review. Information Technology & Libraries* [en ligne]. Décembre 2018. Vol. 37, n°4, pp. 9–14. [Consulté le 28 août 2019]. Disponible à l'adresse :

http://search.ebscohost.com/login.aspx?direct=true&db=lih&AN=133718523&site=ehost-live

BRENER, Nathan E., IYENGAR, S. S. et PIANYKH, O. S., 2005. A conclusive methodology for rating OCR performance. *Journal of the American Society for Information Science & Technology* [en ligne]. Juillet 2005. Vol. 56, n°12, pp. 1274–1287. [Consulté le 28 août 2019]. Disponible à l'adresse : http://search.ebscohost.com/login.aspx?direct=true&db=lih&AN=18172083&site=ehost-live

BREUEL, Thomas M., 2007. Announcing the OCRopus Open Source OCR system. *Google developpers* [en ligne]. 9 avril 2007. [Consulté le 9 janvier 2020]. Disponible à l'adresse : https://developers.googleblog.com/2007/04/announcing-ocropus-open-source-ocr.html

BURGY, Florence, GERSON, Steeve, SCHÜPBACH, Loïc, 2020. *Ex imagine ad litteras* : résultats actuels et espoirs futurs. *Recherche d'IdéeS* [en ligne]. 3 mars 2020. [Consulté le 30 mars 2020]. Disponible à l'adresse : https://campus.hesge.ch/blog-master-is/ex-imagine-ad-litteras-resultats-actuels-et-espoirs-futurs/

CARRASCO, Rafael C., 2014. An Open-source OCR Evaluation Tool. In : *Proceedings of the First International Conference on Digital Access to Textual Cultural Heritage, Madrid, 19-20 mai 2014* [en ligne]. New York : ACM. 2014. pp. 179–184. [Consulté le 28 août 2019]. Disponible à l'adresse : https://dl.acm.org/citation.cfm?doid=2595188.2595221

CIMON, Lucas, 2019. ImproveQuality. *Tesseract Wiki* [en ligne]. 25 novembre 2019. [Consulté le 9 janvier 2020]. Disponible à l'adresse : https://github.com/tesseract-ocr/tesseract/wiki/ImproveQuality

CLEMATIDE, Simon, FURRER, Lenz et VOLK, Martin, 2016. Crowdsourcing an OCR Gold Standard for a German and French Heritage Corpus. In : *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016), Portorož, 23-28 mai 2016* [en ligne]. 2016. pp. 975-982. [Consulté le 28 août 2019]. Disponible à l'adresse : https://www.zora.uzh.ch/id/eprint/124786

COJOCARU, Svetlana et al., 2016. Optical Character Recognition Applied to Romanian Printed Texts of the 18th–20th Century. *Computer Science Journal of Moldova* [en ligne]. 2016. Vol. 24, n°1 (70), pp. 106-117. [Consulté le 28 août 2019]. Disponible à l'adresse : http://www.math.md/files/csjm/v24-n1/v24-n1-(pp106-117).pdf

DASH, Kalyan S., PUHAN, N. B. et PANDA, G., 2017. Odia character recognition : a directional review. *The Artificial Intelligence Review* [en ligne]. 2017. Vol. 48, n°4, pp. 473–497. [Consulté le 28 août 2019]. Disponible à l'adresse : https://search.proquest.com/lisa/docview/1961506152/abstract/35B11DA70B14444EPQ/2

E-RARA, 2020. *E-rara* [en ligne]. 11 janvier 2020. [Consulté le 11 janvier 2020]. Disponible à l'adresse : https://www.e-rara.ch/

EUROPEANA, 2020. *Europeana Collections* [en ligne]. 11 janvier 2020. [Consulté le 11 janvier 2020]. Disponible à l'adresse : https://www.europeana.eu/

GHOSH, Kripabandhu et al., 2016. Improving Information Retrieval Performance on OCRed Text in the Absence of Clean Text Ground Truth. *Information Processing & Management* [en ligne]. 1 septembre 2016. Vol. 52, n°5, pp. 873–884. [Consulté le 28 août 2019]. Disponible à l'adresse : http://www.sciencedirect.com/science/article/pii/S030645731630036X

HLÁDEK, Daniel et al., 2017. Learning string distance with smoothing for OCR spelling correction. *Multimedia Tools and Applications* [en ligne]. Novembre 2017. Vol. 76, n°22,

pp. 24549–24567. [Consulté le 28 août 2019]. Disponible à l'adresse : http://link.springer.com/10.1007/s11042-016-4185-5

IIIF, 2019. *IIIF | International Image Interoperability Framework* [en ligne]. 18 décembre 2019. [Consulté le 11 janvier 2020]. Disponible à l'adresse : https://iiif.io/

IIIF, 2020. Community. *IIIF | International Image Interoperability Framework* [en ligne]. 11 janvier 2020. [Consulté le 11 janvier 2020]. Disponible à l'adresse : https://iiif.io/community/#participating-institutions

IMPACT, 2013. *IMPACT Centre of Competence* [en ligne]. 2013. Mis à jour le 9 janvier 2020. [Consulté le 9 janvier 2020]. Disponible à l'adresse : https://www.digitisation.eu/

JÄRVELIN, Anni et al., 2016. Information retrieval from historical newspaper collections in highly inflectional languages: a query expansion approach. *Journal of the Association for Information Science & Technology* [en ligne]. Décembre 2016. Vol. 67, n° 12, pp. 2928–2946. [Consulté le 28 août 2019]. Disponible à l'adresse : http://search.ebscohost.com/login.aspx?direct=true&db=lih&AN=119478036&site=ehost-live

JOST, Clémence, 2019. Lancement d'OCR4all, un outil open source et gratuit de reconnaissance de caractères anciens pour les chercheurs en histoire et les archivistes. *Archimag* [en ligne]. 24 avril 2019. [Consulté le 5 septembre 2019]. Disponible à l'adresse : https://www.archimag.com/archives-patrimoine/2019/04/24/ocr4all-open-source-gratuit-reconnaissance-caracteres-anciens

KANN, Bettina et HINTERSONNLEITNER, Michael, 2015. Volltextsuche in historischen Texten. *Bibliothek Forschung und Praxis* [en ligne]. Avril 2015. Vol. 39, n° 1, pp. 73–79. [Consulté le 28 août 2019]. Disponible à l'adresse : https://www.degruyter.com/downloadpdf/j/bfup.2015.39.issue-1/bfp-2015-0004/bfp-2015-0004.pdf

KARPINSKI, R., LOHANI, D. et BELAÏD, A., 2018. Metrics for Complete Evaluation of OCR Performance. In : *IPCV'18 - The 22nd Int'l Conf on Image Processing, Computer Vision, & Pattern Recognition, Las Vegas, juillet 2018* [en ligne]. 2018. pp. 23-29. [Consulté le 28 août 2019]. Disponible à l'adresse : https://csce.ucmss.com/cr/books/2018/LFS/CSREA2018/IPC3481.pdf

KESTEMONT, Mike, CHRISTLEIN, Vincent et STUTZMANN, Dominique, 2017. Artificial Paleography: Computational Approaches to Identifying Script Types in Medieval Manuscripts. *Speculum* [en ligne]. 2 octobre 2017. Vol. 92, S1, pp. S86–S109. [Consulté le 28 août 2019]. Disponible à l'adresse : https://www.journals.uchicago.edu/doi/10.1086/694112

KISSOS, Ido et DERSHOWITZ, Nachum, 2016. OCR Error Correction Using Character Correction and Feature-Based Word Classification. In : *12th IAPR Workshop on Document Analysis Systems (DAS), Santorini, 11-14 avril 2016* [en ligne]. Avril 2016. pp. 198–203. [Consulté le 28 août 2019]. Disponible à l'adresse : http://ieeexplore.ieee.org/document/7490117/

KUMAR, Munish et al., 2018. Character and numeral recognition for non-Indic and Indic scripts : a survey. *The Artificial Intelligence Review* [en ligne]. 2018. pp. 1–27. [Consulté le 28 août 2019]. Disponible à l'adresse : https://search.proquest.com/lisa/docview/1984338483/abstract/35B11DA70B14444EPQ/1

LEBARBÉ, Thomas et al., 2014. *Les manuscrits de Stendhal en ligne* [en ligne]. 2014. Mis à jour le 11 janvier 2020. [Consulté le 11 janvier 2020]. Disponible à l'adresse : http://stendhal.demarre-shs.fr/

MEI, Jie et al., 2018. Statistical learning for OCR error correction. *Information Processing & Management* [en ligne]. 1 novembre 2018. Vol. 54, n° 6, pp. 874–887. [Consulté le 28 août 2019]. Disponible à l'adresse : http://www.sciencedirect.com/science/article/pii/S0306457317307823

MIRADOR, 2020. *Mirador* [en ligne]. 11 janvier 2020. [Consulté le 11 janvier 2020]. Disponible à l'adresse : https://projectmirador.org/

MORI, Shunji, SUEN, Ching Y. et YAMAMOTO, Kazuhiko, 1992. Historical review of OCR research and development. *Proceedings of the IEEE* [en ligne]. Juillet 1992. Vol. 80, n° 7, pp. 1029-1058. [Consulté le 18 décembre 2019]. Disponible à l'adresse : http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=156468&isnumber=4050

MUEHLBERGER, Guenter et al., 2019. Transforming scholarship in the archives through handwritten text recognition: *Transkribus* as a case study. *Journal of Documentation* [en ligne]. 24 juillet 2019. [Consulté le 28 août 2019]. Disponible à l'adresse : https://www.emeraldinsight.com/doi/10.1108/JD-07-2018-0114

NAGY, George, 2016. Disruptive developments in document recognition. *Pattern Recognition Letters* [en ligne]. 1 août 2016. Vol. 79, pp. 106–112. [Consulté le 28 août 2019]. Disponible à l'adresse : http://www.sciencedirect.com/science/article/pii/S0167865515004109

PTUCHA, Raymond, et al., 2019. Intelligent character recognition using fully convolutional neural networks. *Pattern Recognition* [en ligne]. Avril 2019. Vol. 88, pp. 604–613. [Consulté le 28 août 2019]. Disponible à l'adresse : http://www.sciencedirect.com/science/article/pii/S0031320318304370

REDDY, Sravana et CRANE, Gregory, 2006. A Document Recognition System for Early Modern Latin. In: *Chicago Colloquium on Digital Humanities and Computer Science: What Do You Do With A Million Books* [en ligne]. 2006. Vol. 23, pp. 1-4. [Consulté le 28 août 2019]. Disponible à l'adresse : https://dl.tufts.edu/concern/pdfs/kd17d4036

REHMAN, Amjad et SABA, Tanzila, 2014. Neural networks for document image preprocessing: state of the art. *The Artificial Intelligence Review* [en ligne]. 2014. Vol. 42, n° 2, pp. 253–273. [Consulté le 28 août 2019]. Disponible à l'adresse : https://search.proquest.com/lisa/docview/1542796407/abstract/EC9B8EB6A5EF463APQ/41

RICE, Stephen V., JENKINS, Frank R. et NARTKER, Thomas A., 1996. The Fifth Annual Test of OCR Accuracy. *Information Science Research Institute* [en ligne]. 1996. pp. 1-46. [Consulté le 28 août 2019]. Disponible à l'adresse : http://stephenvrice.com/images/AT-1996.pdf

RONALLO, Jason, 2018. Content Search API. *IIIF Technical Workshop* [en ligne]. 18 mai 2018. [Consulté le 11 janvier 2020]. Disponible à l'adresse : http://ronallo.com/iiif-workshop-new/content-search-api.html

REUL, Christian, 2020. @chreul. thx for the hint and sorry […]. *line segmentation hangs on empty pages · Issue #45* [en ligne]. 13 janvier 2020. [Consulté le 14 janvier 2020]. Disponible à l'adresse : https://github.com/OCR4all/OCR4all/issues/45

RYDBERG-COX, Jeffrey A., 2003. Automatic Disambiguation of Latin Abbreviations in Early Modern Texts for Humanities Digital Libraries. In : *Proceedings of the 3rd ACM/IEEE-CS Joint Conference on Digital Libraries, Houston, 27-31 mai 2003* [en ligne]. Washington DC: IEEE Computer Society. 2003. pp. 372–373. [Consulté le 28 août 2019]. Disponible à l'adresse : http://dl.acm.org/citation.cfm?id=827140.827207

SABER, Shimaa et al., 2016. Performance Evaluation of Arabic Optical Character Recognition Engines for Noisy Inputs. In : Gaber T., Hassanien A., El-Bendary N. et Dey N. *The 1st International Conference on Advanced Intelligent System and Informatics*

(AISI2015), Beni Suef, 28-30 novembre 2015. Cham : Springer, pp. 449-459. [Consulté le 28 août 2019]. Advances in Intelligent Systems and Computing, 407. Disponible à l'adresse : https://link.springer.com/chapter/10.1007/978-3-319-26690-9_40

SASAKI, Yutaka, 2007. The truth oft he F-measure. *Teach Tutor mater* [en ligne]. 26 Octobre 2007. Vol. 1, n° 5, pp. 1-5. [Consulté le 8 décembre 2019]. Disponible à l'adresse : https://www.researchgate.net/publication/268185911_The_truth_of_the_F-measure

SCHANTZ, Herbert F., 1982. *The history of OCR, optical character recognition* [en ligne]. Manchester Center, Vt. : Recognition Technologies Users Association. [Consulté le 18 décembre 2019]. Disponible à l'adresse : https://archive.org/details/historyofocropti0000scha

SEIDMAN, Max J., et al., 2016. Are games a viable solution to crowdsourcing improvements to faulty OCR ? - The Purposeful Gaming and BHL experience. *Code4Lib Journal* [en ligne]. Juillet 2016. Vol. 33, p. 1. [Consulté le 28 août 2019]. Disponible à l'adresse : http://search.ebscohost.com/login.aspx?direct=true&db=lih&AN=116963678&site=ehost-live

SMITH, Ray, 2007. An overview of the Tesseract OCR engine. In : *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007), Curitiba, Paraná, Brésil, 23-26 septembre 2007* [en ligne]. Septembre 2007. Vol. 2, pp. 629-633. [Consulté le 26 octobre 2019]. Disponible à l'adresse : https://ieeexplore.ieee.org/document/4376991?arnumber=4376991

STUTZMANN, Dominique, 2017. Paléographie : la révolution numérique. *L'Histoire* [en ligne]. Septembre 2017. Vol. 439, p. 30. [Consulté le 28 août 2019]. Disponible à l'adresse : https://www.lhistoire.fr/irht-dans-le-secret-des-manuscrits/paléographie-la-révolution-numérique

SUN, Wei et al., 1992. Intelligent OCR Processing. *Journal of the American Society for Information Science* [en ligne]. Juillet 1992. Vol. 43, n°6, pp. 422–431. [Consulté le 28 août 2019]. Disponible à l'adresse : http://search.ebscohost.com/login.aspx?direct=true&db=lih&AN=16918942&site=ehost-live

THERAYSMITH [pseudonyme], 2017. The text corpus is from *all* the www, […]. *Q&A : Indic - length of the compressed codes · Issue #654* [en ligne]. 23 janvier 2017. [Consulté le 22 décembre 2019]. Disponible à l'adresse : https://github.com/tesseract-ocr/tesseract/issues/654#issuecomment-274574951

TUMBE, Chinmay, 2019. Corpus linguistics, newspaper archives and historical research methods. *Journal of Management History* [en ligne]. 30 mai 2019. [Consulté le 28 août 2019]. Disponible à l'adresse : https://www.emeraldinsight.com/doi/10.1108/JMH-01-2018-0009

UNIVERSAL VIEWER, 2019. *Universal Viewer* [en ligne]. 30 août 2019. [Consulté le 11 janvier 2020]. Disponible à l'adresse : http://universalviewer.io/

UNIVERSITY OF CAMBRIDGE, 2015. *Cambridge Digital Library* [en ligne]. 2015. Mis à jour le 11 janvier 2020. [Consulté le 11 janvier 2020]. Disponible à l'adresse : https://cudl.lib.cam.ac.uk/

URIELI, Assaf et VERGEZ-COURET, Marianne, 2013. Jochre, océrisation par apprentissage automatique : étude comparée sur le yiddish et l'occitan. In : *TALARE 2013 : Traitement automatique des langues régionales de France et d'Europe, Les Sables d'Olonne, juin 2013* [en ligne]. 21 juin 2013. pp. 221-234. [Consulté le 28 août 2019]. Disponible à l'adresse : https://hal-univ-tlse2.archives-ouvertes.fr/hal-00979665

VOBL, Thorsten et al., 2014. PoCoTo - an Open Source System for Efficient Interactive Postcorrection of OCRed Historical Texts. In: *Proceedings of the First International Conference on Digital Access to Textual Cultural Heritage, Madrid, 19-20 mai 2014* [en ligne]. New York : ACM. 2014. pp. 57–61. [Consulté le 28 août 2019]. Disponible à l'adresse : https://dl.acm.org/citation.cfm?id=2595197

WICK, Christoph, REUL, Christian et PUPPE, Frank, 2018. Calamari - A High-Performance Tensorflow-based Deep Learning Package for Optical Character Recognition. [En ligne]. Preprint. 6 août 2018. [Consulté le 09 janvier 2020]. Disponible à l'adresse : https://arxiv.org/abs/1807.02004

ZHOU, Yongli, 2010. Are Your Digital Documents Web Friendly? : Making Scanned Documents Web Accessible. *Information Technology & Libraries* [en ligne]. Septembre 2010. Vol. 29, n°3, pp. 151–160. [Consulté le 28 août 2019]. Disponible à l'adresse : http://search.ebscohost.com/login.aspx?direct=true&db=lih&AN=52871764&site=ehost-live