UNIVERSITE LUMIERE LYON 2

Rapport de stage

Master 1 Informatique

Parcours du master

Institut de la Communication

Expérimentation et évaluation d'outils d'OCR et d'OLR

Edina ADJARO PATOUSSI

Encadrant :

Ludovic MONCLA,
Julien VELCIN

Enseignant encadrant :

Valentin Lachand-Pascal

2023 - 2024

RESUME

Dans ce document, nous avons **expérimenté** et **évalué** le rendu de certains outils **de segmentation de documents** et de **reconnaissance de caractères** sur des versions du **dictionnaire de Trévoux de 1704 et de 1743**. Pour ce faire, dans un premier temps, nous avons testé différents outils qui nous ont été proposés et nous avons recensé les plus pertinents répondant à notre problématique. Ensuite, nous avons proposé une méthode permettant **l'amélioration continue** d'un modèle de **deep learning PrimaLayout basé sur Detectron2 de Facebook**. Enfin, nous avons étudié les métriques qui pourraient nous permettre d'évaluer la capacité de ces outils à bien segmenter nos documents et à correctement océriser nos textes.

MOTS-CLES

OLR ou Segmentation : Découpage de l'image en zones d'information, potentiellement à plusieurs niveaux différents : paragraphe, article, vedette, sous-vedette

OCR : Reconnaissance de caractères

Trévoux : Dictionnaire ancien français qui date du 19ème siècle

Vedette : Dans le contexte du dictionnaire de notre étude, une vedette est un mot tout en majuscules qui sera défini.

Sous-vedette : Dans le contexte du dictionnaire de notre étude, une sous-vedette est une vedette mais avec seulement une partie de ses lettres en majuscules et qui est liée elle-même à une vedette.

Article : Dans le contexte du dictionnaire de notre étude, un article constitue toute la définition d'une vedette.

SOMMAIRE

INTRODUCTION GENERALE

Contexte

Ce stage s'inscrit dans le cadre du projet Trévoux, qui a pour but de numériser les pages scannées du Dictionnaire de Trévoux tout en préservant sa structure interne. L'objectif est de produire un format numérique exploitable par les linguistes, facilitant ainsi l'analyse et l'étude du contenu.

Les travaux réalisés dans ce cadre constituent une étape préliminaire au projet GEODE, dont les objectifs sont :

- Étudier l'évolution sémantique à travers les différentes versions du Dictionnaire de Trévoux.
- Classifier les domaines de connaissance représentés dans le dictionnaire.

La numérisation des pages du dictionnaire est une étape cruciale pour atteindre ces objectifs. C'est dans cette perspective que le projet Trévoux a été financé par le Labex ASLAN, un instrument du programme d'investissements d'avenir. Ce programme vise à :

- Augmenter l'excellence et l'originalité scientifique, favoriser le transfert des connaissances et renforcer l'attractivité internationale de la recherche française, tout en impliquant d'autres laboratoires nationaux.
- Garantir l'excellence pédagogique et jouer un rôle moteur dans les formations de niveau master et doctorat.
- S'inscrire dans la stratégie de ses établissements de tutelle et renforcer la dynamique des sites concernés.

Ce stage se concentre sur l'expérimentation et l'évaluation d'outils de reconnaissance de caractères (OCR) et de reconnaissance de contours (OLR), appliqués aux éditions du Trévoux de 1704 et 1743.

Objectif

Ce stage a pour objectif de trouver les meilleurs outils ou combinaisons d'outils de segmentation et d'OCR capables de réaliser le moins d'erreurs possible sur les pages du Trevoux. Ensuite, il s'agit de qualifier et quantifier les résultats obtenus en les comparant à un échantillon de vérité terrain de 1743 dont nous disposons.

Pour ce faire, les étapes suivantes seront réalisées :

- Tester et comparer les **outils d'OLR/OCR** listés ci-dessous sur les pages **de Trevoux de 1704 et 1743** :
  - Tesseract
  - Layout Parser
  - Surya
  - Grobid
  - Laypa

- **Constituer un jeu de données annoté** pour l'entraînement de modèles d'OLR/OCR avec Label Studio ou Prodigy si nécessaire.
- Développer une méthode permettant de comparer la sortie de ces outils avec la vérité terrain.

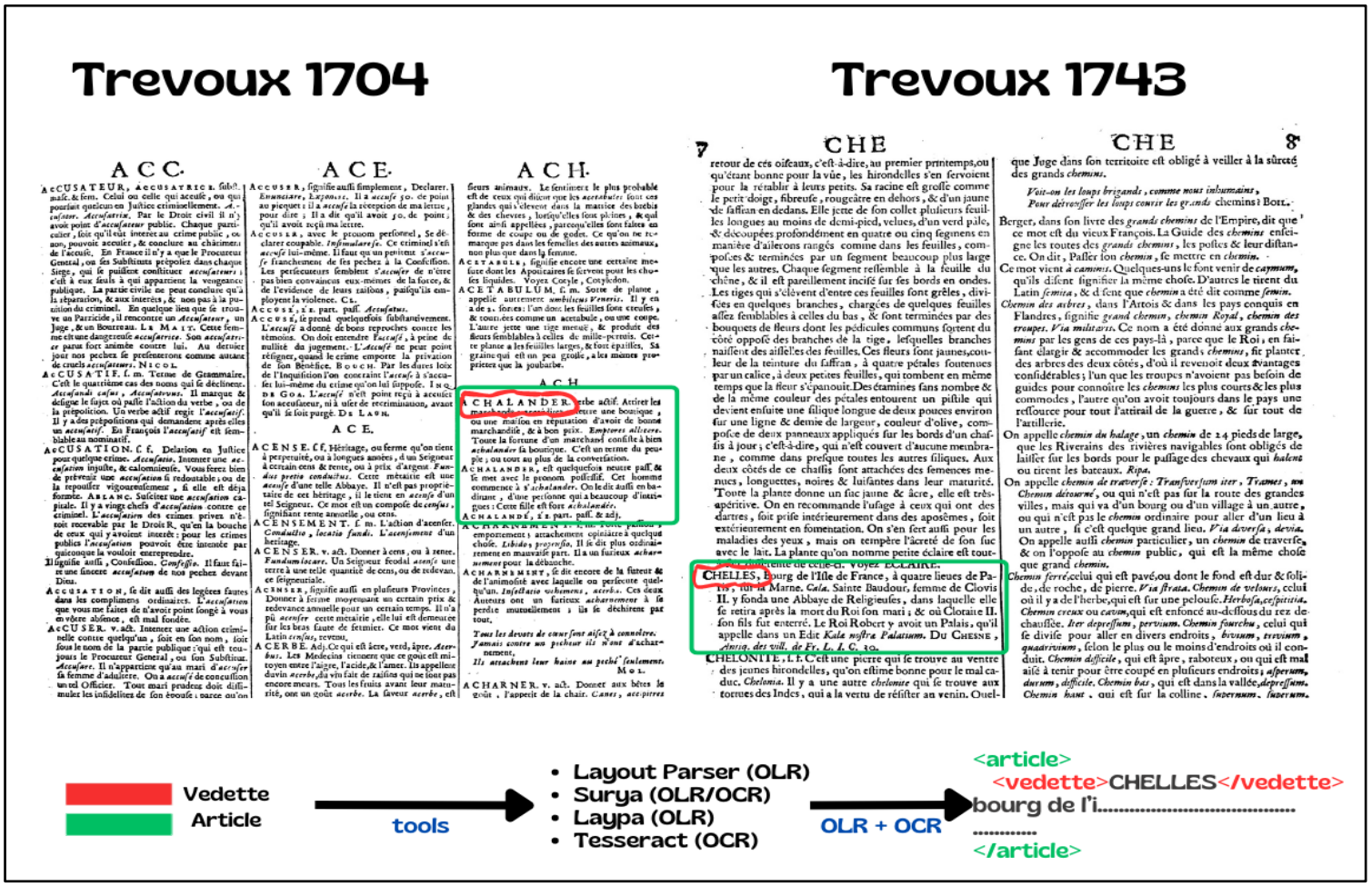

Figure 1 : illustration de l'objectif

Segmentation ou OLR

Définition

Suivant la definition de Wikipédia, La **segmentation d'image** est une opération de traitement d'images consistant à détecter et rassembler les pixels suivant des critères, notamment d'intensité ou spatiaux, l'image apparaissant ainsi formée de régions uniformes.

Actuellement, on dispose de plusieurs Framework qui réalisent la segmentation d'image. Parmi les plus célèbres, on trouve :

- YOLOv5, développé sur PyTorch, connu pour sa rapidité et sa précision, particulièrement adapté à la détection en temps réel.

- Detectron2, développé par Facebook, également construit sur PyTorch, réputé pour sa flexibilité et ses performances élevées dans la détection.

Pourquoi :

Dans notre projet, l'étape de segmentation fait partie des étapes les plus importantes car elle nous permettra de :

- **Bien Extraire la Structure des Dictionnaires :** Cela est très important pour les linguistes car une segmentation précise permet de capturer fidèlement la structure des dictionnaires
- **Améliorer la Précision de l'OCR :** Une segmentation réussit permettra aux OCR de fonctionner plus efficacement.

Erreur de Segmentation :

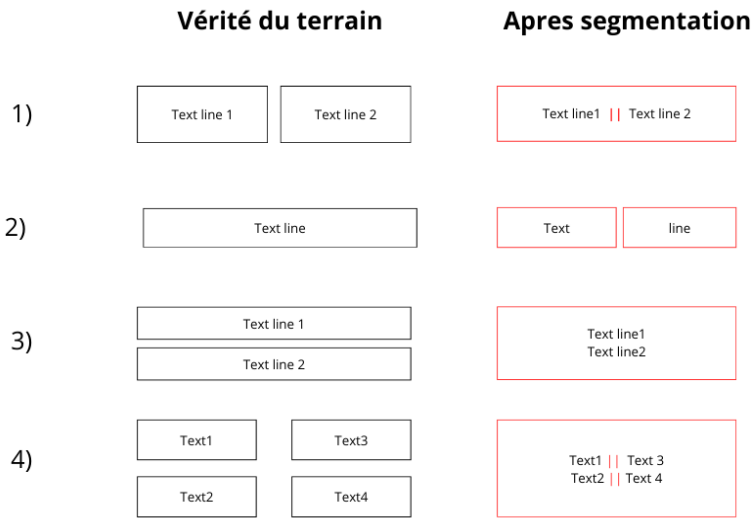

Figure 2 : illustration des erreurs de segmentation récurrent

Généralement après un processus de segmentation, on a **4** types d'erreur :

- **Erreur de fusion horizontale :** c'est quand notre **OLR** concatène ou sépare deux blocs de texte horizontalement, comme dans les exemples **(1), (2), (3)** dans la figure ci-dessus.
- **Erreur de fusion verticale :** c'est quand notre **OLR** concatène ou sépare deux blocs de texte verticalement, comme dans les exemples **(3), (4)** dans la figure ci-dessus.

Reconnaissance de caractères (OCR)

Définition

Suivant[1] : La reconnaissance de caractères consiste à identifier la forme à partir d'image.

Certaines méthodes ne vont pas reconnaître les caractères indépendamment les uns des autres mais au contraire, reconnaître des mots entiers

Dans l'état actuel des choses, la plupart des outils d'OCR détectent une ligne (line detection), comme illustré à la figure 7. Pour extraire des structures complexes d'une page, il est nécessaire de passer d'abord par une reconnaissance optique des lettres (OLR) avant d'effectuer l'OCR.

Erreur :

Les erreurs de reconnaissance se situent donc au niveau des caractères.
On distingue 4 types d'erreurs :

- **Délétion** : caractère manquant
  word −→ wrd
- **Substitution** : un caractère remplacé par un autre (voire plusieurs)
  word −→ w0rd word −→ ivord
- **Insertion** : caractère inséré dans le texte
  word −→ wordsd

La plupart des erreurs de reconnaissance de caractères proviennent d'une mauvaise interprétation de la morphologie des caractères à reconnaître. Ces erreurs vont beaucoup plus s'accentuer du fait que nous avons des pages qui proviennent d'une ancienne version du français que la plupart des OCR ne connaissent pas.

[1] Karpinski et Belaid, « Rapport Evaluation des OCR ».

EXPERIMENTATION DES OUTILS

Layout Parser

Layout Parser est une bibliothèque Python qui permet de faire de l'analyse d'image en détectant les **layouts** (les contours) des différents éléments d'une image basée sur **Detectron-2.**
Pour l'utiliser, il y a deux moyens de procéder :

- Utiliser des modèles existants.
- Construire soi-même un modèle adapté à notre cas.

Modèles existants

Layout parser dispose de 6 principaux modèles adaptés pour différents cas d'utilisation https://layout-parser.readthedocs.io/en/latest/notes/modelzoo.html . Pendant les tests, j'ai utilisé 4 modèles qui semblent être les plus proches de notre problématique

- **HJDataset** : Adapté pour les documents historiques, permet de détecter des parties :

MAP: {1:"Page Frame", 2:"Row", 3:"Title Region", 4:"Text Region", 5:"Title", 6:"Subtitle", 7:"Other"}

Résultat obtenu :

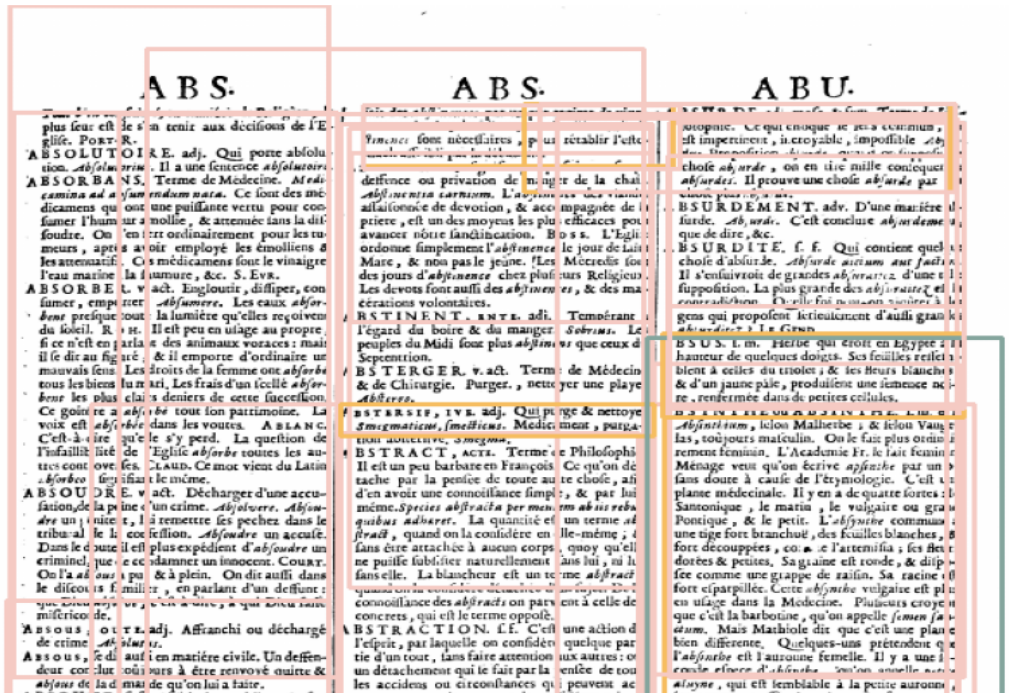

*Figure 3: résultat du **HJDataset***

- **PubLayNet** : Modèle adapté pour les publications scientifiques.

  MAP: {0: "Text", 1: "Title", 2: "List", 3:"Table", 4:"Figure"}

Résultat obtenu :

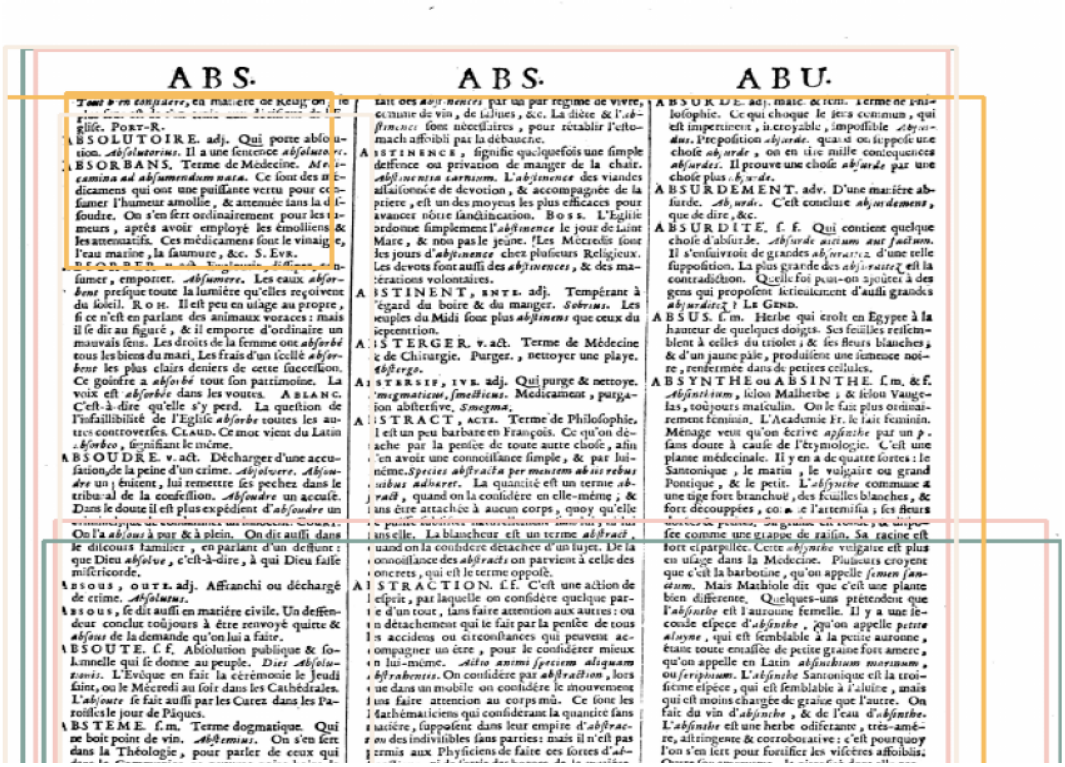

Figure 4: résultat du PubLayNet

- **Newspaper Navigator** : Modèle adapté pour la presse.

  MAP : {0: "Photograph", 1: "Illustration", 2: "Map", 3: "Comics/Cartoon", 4: "Editorial Cartoon", 5: "Headline", 6: "Advertisement"}

Résultat obtenu : En utilisant ce modèle, nous n'avons pas obtenu de résultats en ayant une détection sur les pages.

- **PrimaLayout:**

  MAP: {1:"TextRegion", 2:"ImageRegion", 3:"TableRegion", 4:"MathsRegion", 5:"SeparatorRegion", 6:"OtherRegion"}

Résultat obtenu :

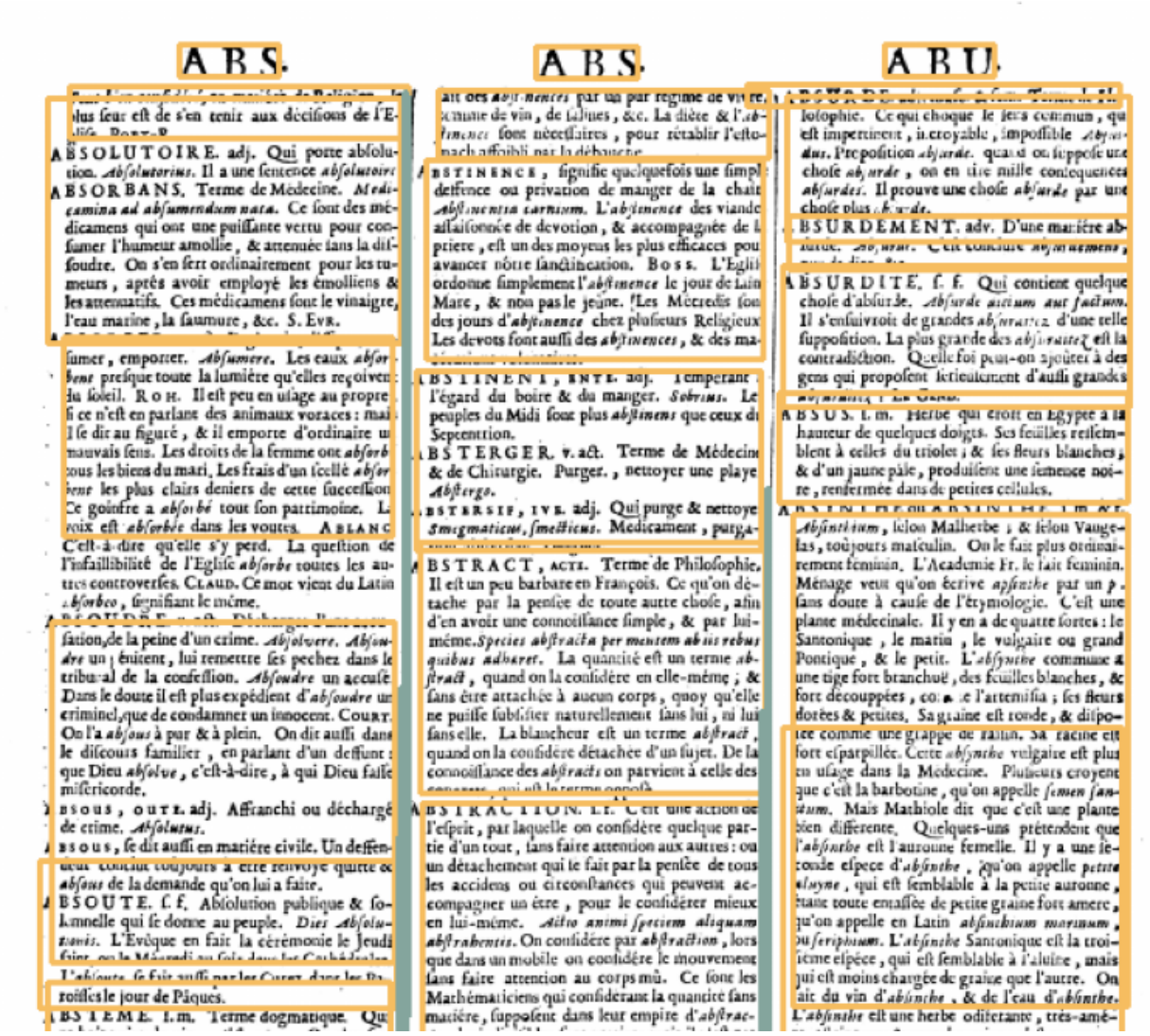

Figure 5: résultats du PrimaLayout

Laypa :

Laypa est un outil de segmentation dont le but est de trouver des régions (paragraphe, numéro de page, etc.) et des lignes de base dans les documents.
Les modèles sont construits à l'aide du Framework detectron2 .
Les lignes de base et les classifications de région sont ensuite mises à disposition pour un traitement ultérieur.

Résultat obtenu :

Figure 6: Resultat avec laypa

Inconvénient

La production de datasets de training pour entraîner Laypa semble difficile car elle nécessite des images avec des masques.

Grobid

GROBID est une qui permet d'analyser la structure d'un document, d'extraire le texte d'un document et de le restituer sous forme d'un fichier XML/TEI.

Avantages

- GROBID dispose d'une **interface graphique** intuitive, donc simple à utiliser.
- GROBID offre la possibilité **d'entraîner** le modèle de GROBID https://grobid.readthedocs.io/en/latest/Training-the-models-of-Grobid/
- GROBID offre la possibilité d'avoir le rendu de l'analyse aussi en **TEI/XML**

Inconvénients

- GROBID n'a pas respecté l'ordre des textes et des parties dans le fichier TEI/XML qu'il a généré.
- La production des dataset d'entrainement semble difficile et couteux

Tesseract

Tesseract est une librairie qui permet de faire de l'extraction de texte **OCR**.
Pour l'utiliser sous Python, on peut passer par le package PyTesseract.

Avantages

- Très performant dans la reconnaissance des blocs de texte.
- Offre la possibilité d'exporter le résultat sous différents formats **txt, hocr**.
- Tesseract est open source.

Inconvénients

- Tout comme Surya, il a des difficultés à reconnaître certains caractères comme le **S-long.**
- N'offre pas la possibilité de faire de l'apprentissage.

Résultat obtenu

Figure 8 : résultat OCR avec tesseract

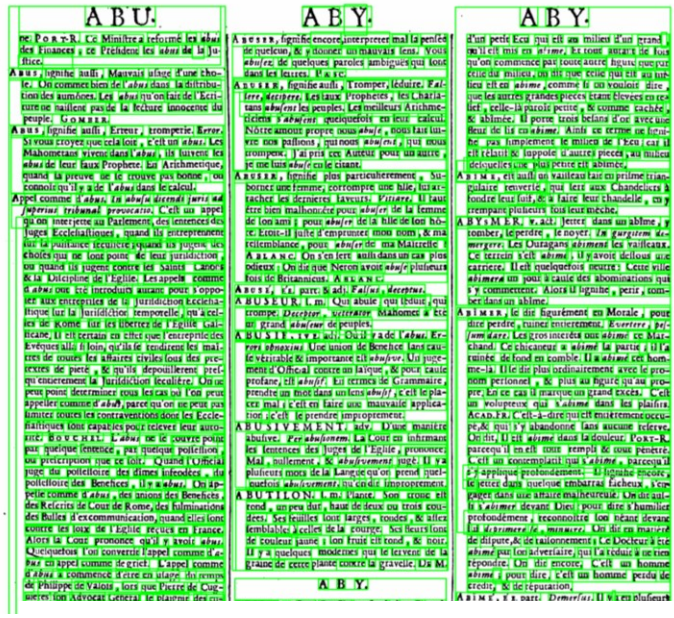

Figure 7: résultat segmentation avec tesseract

Surya

Surya est un outil Python qui permet de réaliser la reconnaissance de caractères **OCR** et la détection de contour **OLR**.

Avantages

- Surya permet de faire à la fois la détection de caractères **OCR** et la détection de layout.
- Surya permet de récupérer le rendu de l'analyse au format **JSON** ou **HOCR** (assez semblable au JSON).
- **Open source**.
- Encore en phase de développement.

Inconvénients

- Surya donne l'impression de fonctionner comme une **boîte noire** et ne donne **pas la possibilité d'améliorer son modèle**
- Pendant l'OCR, Surya a du mal à reconnaître le **s long**.
- Pendant la détection de layout, Surya reconnaît parfois les lignes des colonnes comme le caractère `|` .
- Surya n'a pas une documentation exhaustive.

Résultat obtenu :

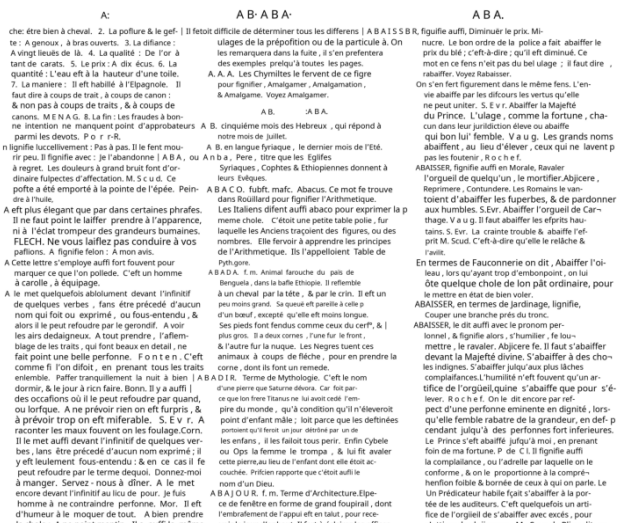

Figure 10: résultat OCR surya

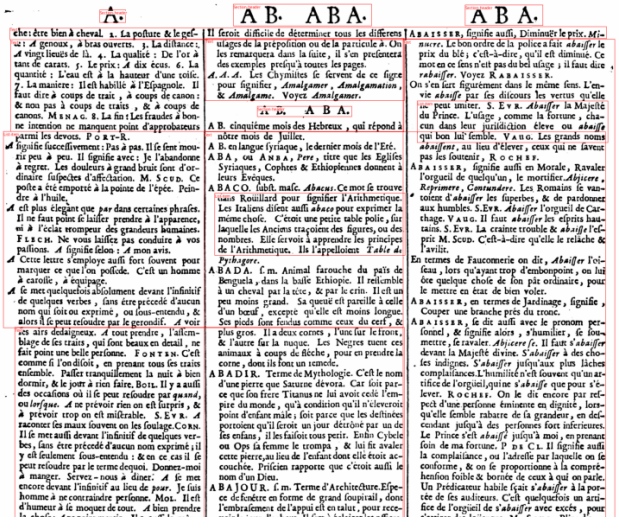

Figure 9: résultat OLR surya

Tableau Récapitulatif :

	Documentation	Type-Input	Type-Output	Open source	Fine -Tunning
Layout Parser	Exhaustive	Image, PDF	JSON , CSV	Oui	Oui
Laypa	Peux exhaustive	Image	XML	Oui	Possible mais difficile de constituer le dataset d'entrainement
Grobid	Exhaustive	PDF	JSON , XML	Oui	Possible mais difficile de constituer le dataset d'entrainement
Tesseract	Exhaustive	Image, PDF	Txt, HOCR	Oui	Non
Surya	Presque inexistant	Image, PDF	JSON , HOCR	Oui	Non

Interprétation :

Au vu de l'expérimentation de ces outils : Layout Parser, Laypa, Grobid, Tesseract, et Surya, nous pouvons conclure que la plupart des outils d'OLR ne répondent que partiellement à notre problématique, qui est d'extraire la structure des pages du Trévoux. Une solution à notre problème pourrait être l'amélioration d'un modèle comme celui de Layout Parser, qui pourrait être fine-tuner, puis de combiner le résultat obtenu avec l'OCR.

PROPOSITION DE METHODOLOGIE

Description de l'approche

Notre approche consiste à affiner le modèle primaModel basé sur Detectron 2, qui fonctionne assez bien sur les pages du Trévoux. Pour ce faire, nous avons défini certaines étapes, qui sont les suivantes :

Etape 1 : Préparation des données :

La création de l'échantillon d'apprentissage se fait en deux étapes :

- **Conversion des pages du Trévoux de 1743 en format image** :
  Convertir les pages du Trévoux de 1743 en images de haute qualité, adaptées pour être utilisées comme entrée dans notre modèle.

- **Sélection d'un échantillon représentatif :**
  Sélectionner les pages qui constitueront un échantillon représentatif des pages du Trévoux de 1743.

Etape 2 : Génération des annotations :

La deuxième étape de notre approche consiste en la génération d'annotations pour les échantillons que nous avons constitués à l'étape 1 à l'aide du modèle. À ce niveau, il existe deux options Possibles :

- **Le modèle Prima :** utilisé pendant la première itération de notre approche.
- **Le modèle Fine-Tuner n-1 :** utilisé pendant l'itération n de notre approche.

Etape 3 : Correction des annotations :

La troisième étape consiste à vérifier si les annotations générées par le modèle sont correctes et à passer à une annotation manuelle si nécessaire. Pour ce faire, nous avons déployé une instance de Label Studio en ligne à l'adresse suivante : https://geode-project-a8f7e0cbcb90.herokuapp.com

Après la correction des annotations, l'ensemble (image et annotations au format JSON) est exporté dans un format **COCO** qui sera exploitable ultérieurement pour réentraîner notre modèle.

Etape 4 : Fine Tuning du model :

La quatrième étape de notre approche consiste au fine-tuning des modèles. Pour ce faire, nous avons deux types d'approches :

- **Entraînement de plusieurs petits modèles :** il s'agit d'entraîner plusieurs sous-modèles spécialisés, chacun cherchant à détecter un seul type de catégorie dans nos pages du Trévoux (sous les vedettes, les sous-vedettes ou les articles), et de combiner ensuite les résultats de tous les sous-modèles dans un fichier JSON.

- **Entraînement d'un grand modèle :** il s'agit d'entraîner un grand modèle capable de détecter tous les types de catégories en même temps (sous les vedettes, les sous-vedettes ou les articles).

Etape 5 : Evaluation du nouveau model et OCR :

L'étape 5 consiste principalement en une évaluation visuelle du modèle que nous avons entraîné pour déterminer s'il est nécessaire ou non de continuer nos itérations. Si le modèle est jugé insatisfaisant, nous pouvons directement recommencer à l'étape 3 en corrigeant les résultats obtenus du modèle. Si le modèle est jugé satisfaisant, l'étape suivante consiste à utiliser l'OCR sur les pages segmentées que nous aurons générées avec notre modèle.

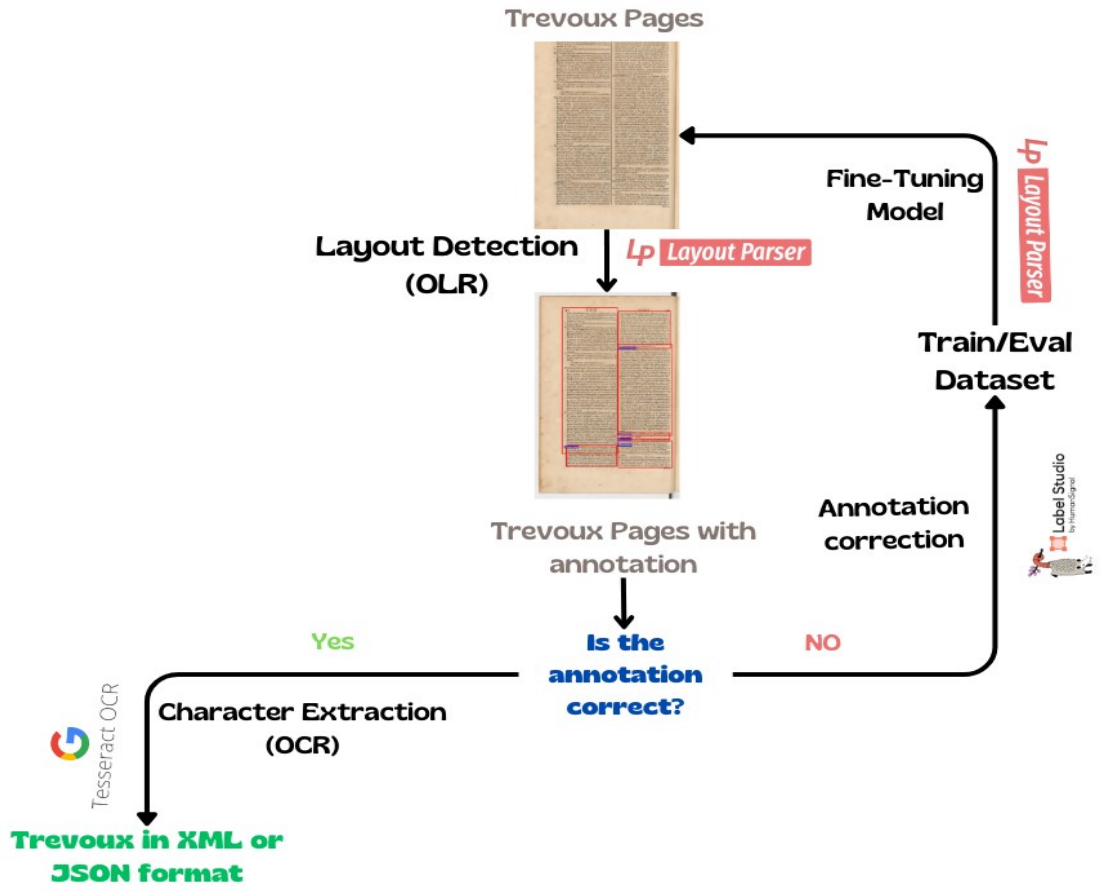

Figure 11: illustration de la méthodologie

Résultats obtenus

Voici une illustration de résultat obtenue :

Avant

Figure 13: résultat obtenue avec le primaModèle

Apres

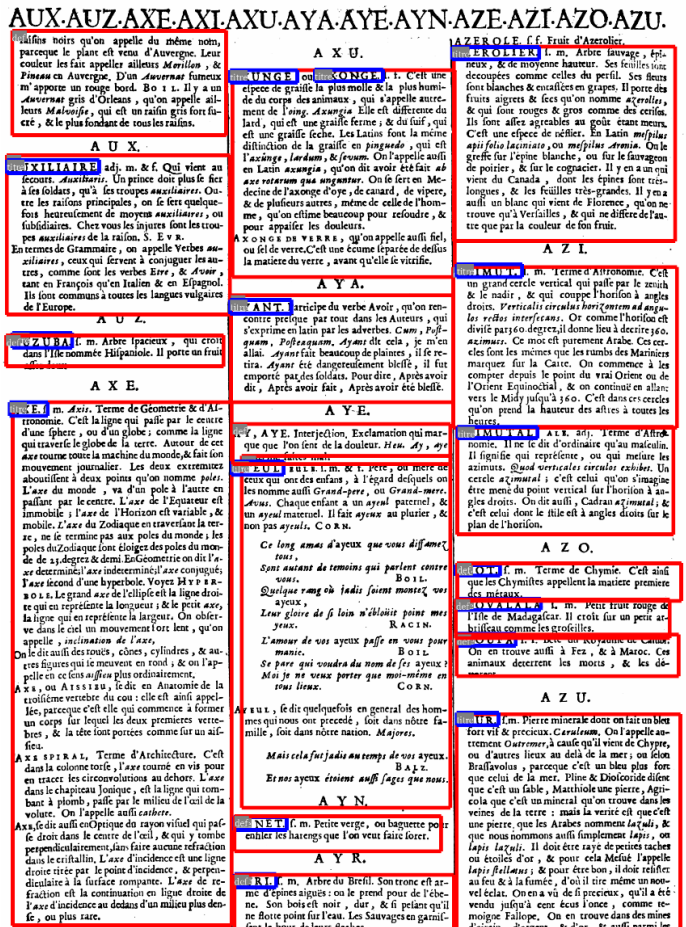

Figure 12: résultat obtenue avec le nouveau modèle

Interprétation :

Cette expérimentation a été réalisée avec une base de données d'entraînement constituée de 100 pages du Trévoux, qui ont été annotées, et de 50 000 itérations.
Le résultat obtenu sur la figure ci-dessus montre que notre modèle entraîné fonctionne assez bien, car nous parvenons désormais à bien détecter les blocs en bleu qui représentent les vedettes et les blocs en rouge qui représentent les articles.
Bien que nous arrivions désormais à mieux détecter les articles et les vedettes, le modèle semble encore commettre des erreurs fréquentes, en oubliant certains blocs et en ratant certains contours de blocs.
Au vu de ces résultats, nous pouvons conclure que notre approche semble efficace, mais qu'une augmentation de la quantité et de la qualité du dataset d'entraînement semble indispensable pour obtenir de meilleurs résultats

ÉVALUATION DES RESULTATS

Évaluation de l'OLR (segmentation)

Pour évaluer la capacité de nos modèles à bien segmenter les pages du Trévoux, nous disposons d'un échantillon de vérité terrain du Trévoux de 1743, que nous avons traité et réorganisé pour pouvoir l'exploiter pour évaluer notre modèle.
Enfin, nous avons calculé les métriques pour évaluer deux modèles que nous avons fine-tuner :

- **XP1 :** Un modèle de détection qui a été fine-tuner sur un dataset combinant la détection de vedettes et d'articles.
- **XP2 :** Un modèle basé sur deux sous-modèles spécialisés, chacun dans la détection de vedettes et d'articles

➢ **Nombre total d'articles détectés par les modèles avec le nombre d'articles réels**

➢ **Nombre total de vedette détectés par les modèles avec le nombre d'articles réels**

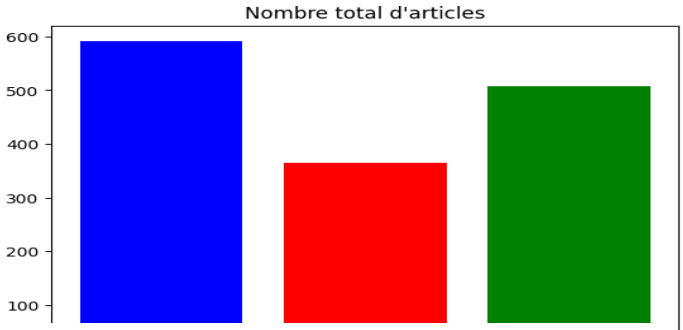

Figure 14 : Nombre total d'articles détectés

Interprétation :

On remarque Nombre total de vedette détectés par les modèles avec le nombre d'articles réels que le modèle XP2 est plus performant que le modèle XP1 en termes de détection d'articles. En effet, le modèle XP1 a détecté 364 articles sur 591 possibles, soit un ratio de 0,615, alors que XP2 a détecté 508 articles sur 591, soit un ratio de 0,859

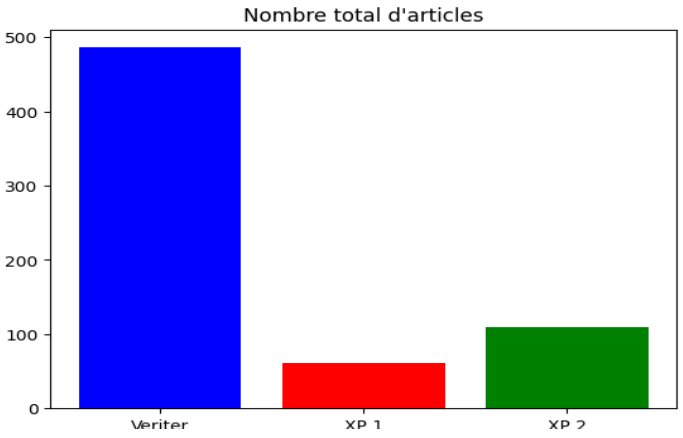

Figure 15:Nombre total de vedette détectés

Interprétation :

On remarque que les deux modèles sont relativement mauvais dans la détection de vedettes. En effet, le modèle XP1 a détecté 61 vedettes sur 486 possibles, soit un ratio de 0,125, alors que XP2 a détecté 110 vedettes sur 486, soit un ratio de 0,22.

➢ **Comptage du nombre d'articles par page égaux, en trop et manquants prédit par les modèles**

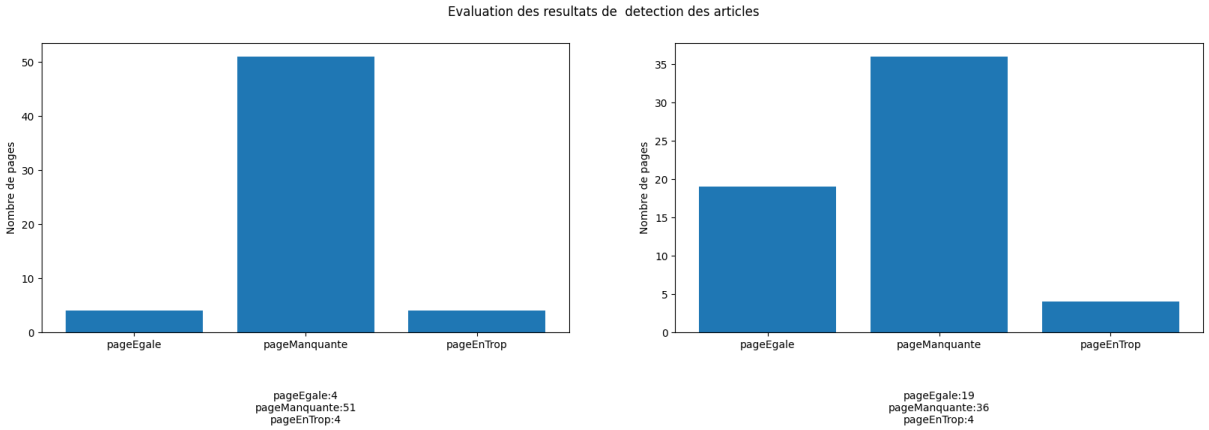

Figure 16: Comptage du nombre d'articles prédit par les modèles

Interprétation :

En zoomant sur le nombre de détections manquantes et en trop par page d'article, on remarque qu'il y a dans la plupart des cas une sous-détection, sauf dans le cas de 4 pages où les 2 modèles font de la surdétection.

➢ **Comptage du nombre de vedette par page égaux, en trop et manquants prédit par les modèles**

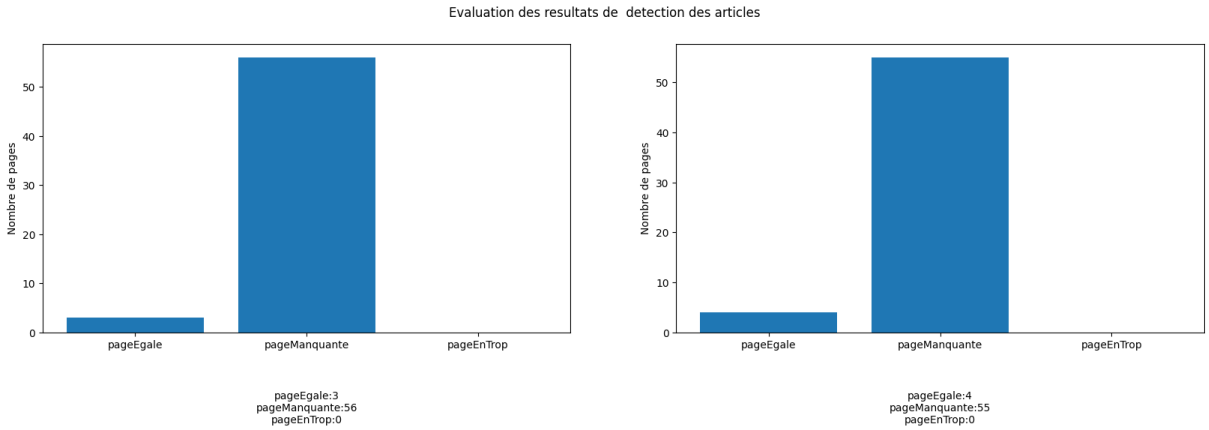

Figure 17:Comptage du nombre de vedette prédit par les modèles

Interprétation :

En zoomant sur le nombre de détections manquantes et en trop par page de vedette, on remarque qu'il y a dans la plupart des cas une sous-détection donc nos 2 modèle ne sont pas assez bonne pour détecter les vedettes

➢ **Différence entre le nombre d'articles prédit par les modèles et le nombre d'articles réels par page**

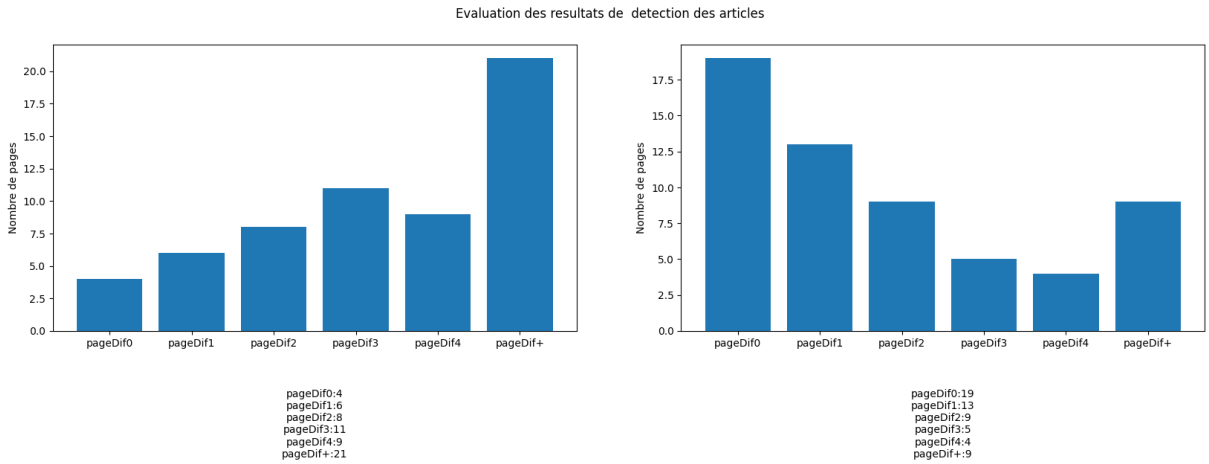

Figure 18:Différence entre le nombre d'articles prédit par les modèles

<u>Interprétation :</u>

Ces graphiques montrent que le modèle XP2 (à droite) est beaucoup plus efficace dans la détection d'articles, car il fait le moins d'erreurs contrairement à XP1.

➢ **Différence entre le nombre de vedette prédit par les modèles et le nombre d'articles réels par page**

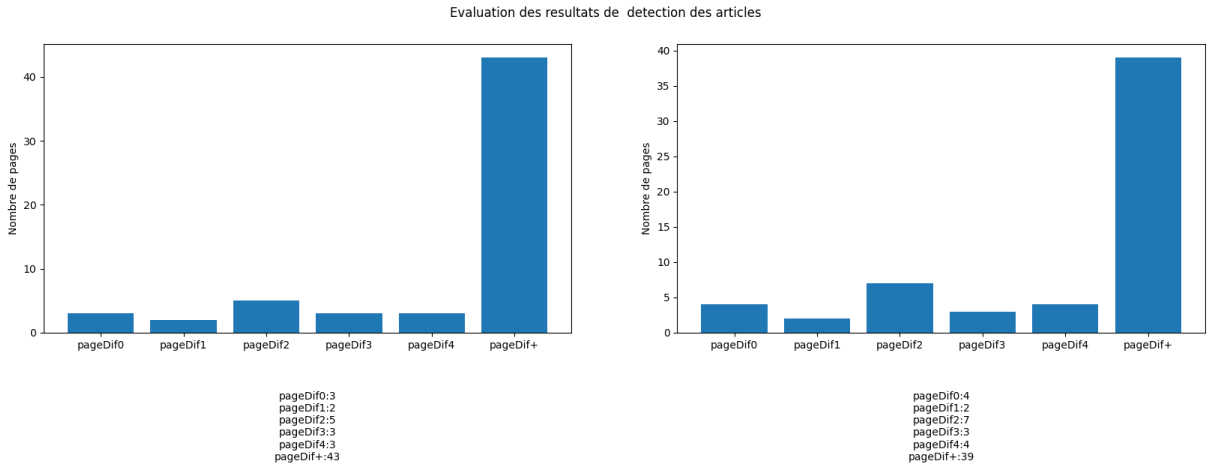

Figure 19: Différence entre le nombre d'articles prédit par les modèles

Interprétation :

Dans la détection de vedettes, il n'y a pas de véritable tendance qui se dégage, car les deux modèles semblent être aussi mauvais l'un que l'autre.

- **Zoom sur les pages avec le plus grand nombre d'erreurs de détection d'articles et de vedettes, et visualisation des prédictions des modèles**

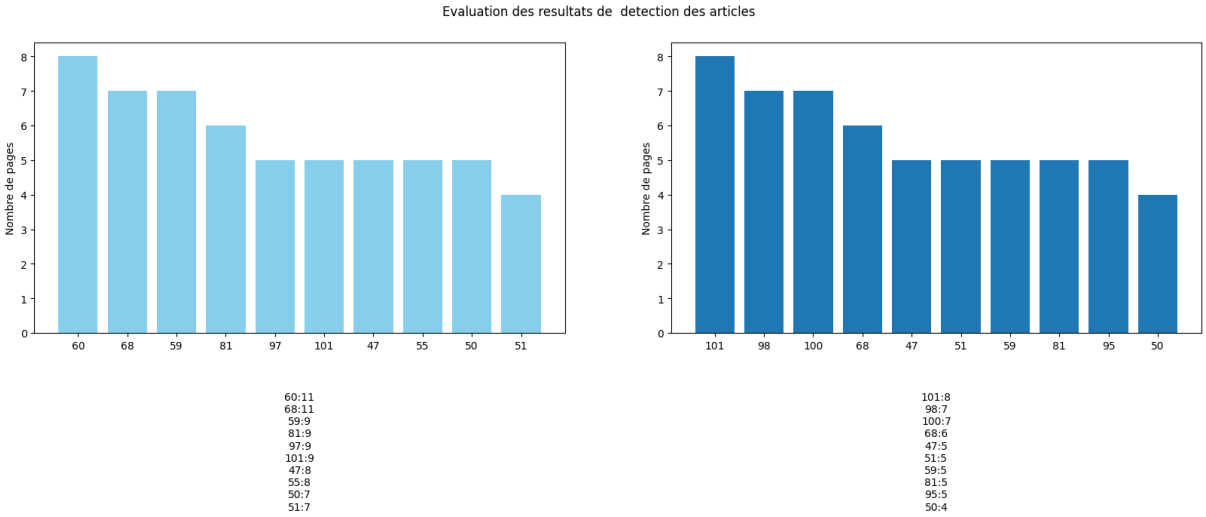

Figure 20: les pages avec le plus grand nombre d'erreurs de détection

Interprétation :

En s'intéressant aux pages où nos modèles ont commis le plus d'erreurs, on remarque que le modèle XP1 enregistre les pages avec le plus d'erreurs. Au-delà de cela, on remarque que sur les pages où le XP2 a commis des erreurs, le XP1 commet aussi des erreurs sur ces mêmes pages avec des taux supérieurs ou égaux au nombre d'erreurs du XP2, par exemple sur les pages 101 et 50.

Nous nous sommes aussi intéressés aux pages où nos modèles ont commis le moins d'erreurs. On remarque que le XP1 réussit uniquement sur des pages au format simple (pas de vedette à détecter et seulement un article sur deux colonnes). Le XP2, contrairement au XP1, arrive à réussir sur des pages avec des structures un peu plus complexes.

Évaluation de l'OCR

La seconde partie de l'évaluation porte sur la reconnaissance des caractères. Pour cela, nous avons effectué des travaux préliminaires visant à reconstituer notre échantillon de vérité terrain et nos prédictions, en les triant par page et par article. Ensuite, nous avons évalué la capacité de notre OCR Tesseract à détecter correctement les caractères.

Analyse par Page :

Pour effectuer une analyse par page, nous avons effectué un travail préliminaire composé des étapes suivantes :

- Réaliser une prédiction avec notre modèle sur un échantillon du Trévoux.
- Utiliser l'OCR Tesseract pour extraire les textes.
- Reconstituer les prédictions par page.
- Reconstituer l'échantillon de vérité terrain par page à l'aide des marqueurs disponibles dans nos fichiers.
- Effectuer une jointure entre les prédictions et la vérité terrain sur la clé des pages
- Réaliser un alignement grâce à l'algorithme RETAS.
- Afficher et comparer les résultats.

Après avoir reconstitué les données par page, voici une illustration d'une page alignée, avec le texte original à gauche et le texte prédit à droite.

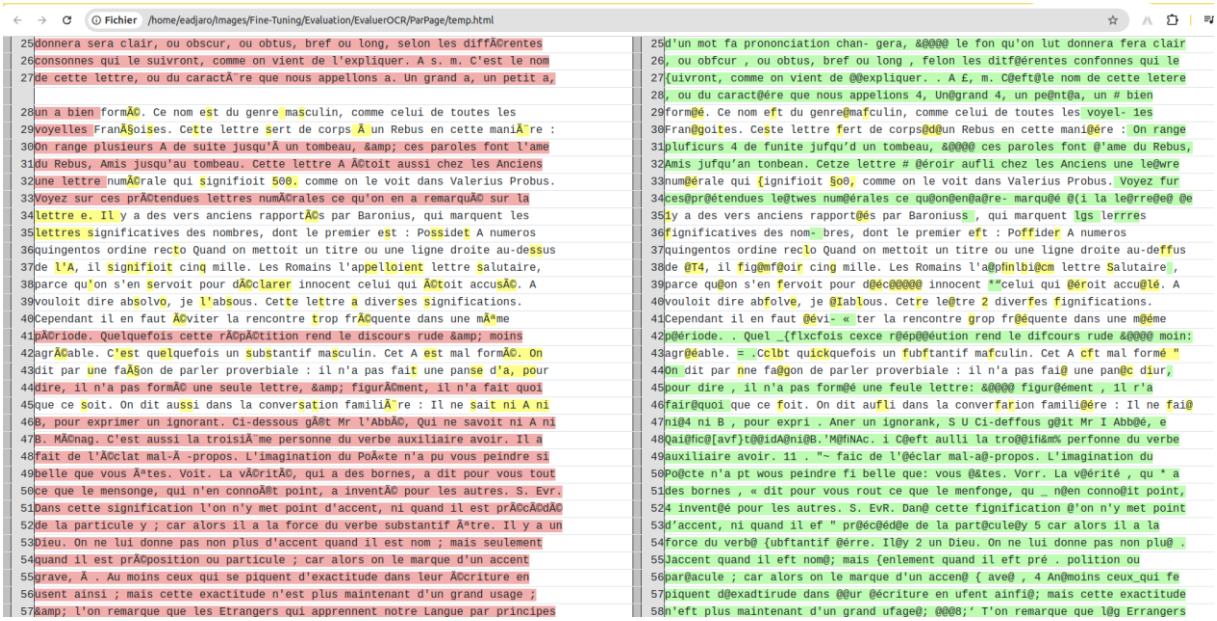

Figure 21 : Résultat obtenu avec le texte original à gauche et le texte prédit à droite

Analyse par Article :

Pour effectuer une analyse par article, nous avons également suivi un travail préliminaire composé des étapes suivantes :

- Réaliser une prédiction avec notre modèle sur un échantillon du Trévoux.
- Utiliser l'OCR Tesseract pour extraire les textes.
- Reconstituer les prédictions par article.
- Reconstituer l'échantillon de vérité terrain par article à l'aide des balises <article> disponibles dans nos fichiers.
- Effectuer une jointure entre les prédictions et la vérité terrain.

En raison de prédictions d'articles manquantes dues à notre modèle, l'étape de jointure entre les prédictions et la vérité terrain n'a pas pu être réalisée.

Cette approche pourrait être utilisée à l'avenir lorsque nous aurons une meilleure précision dans la détection des articles.

Interprétation des résultats :

L'analyse des résultats obtenus par l'analyse par page montre que :

- Les contours prédits par nos modèles ne sont pas toujours parfaits, ce qui entraîne des textes manquants.
- Notre OCR Tesseract a tendance à commettre beaucoup d'erreurs, comme la confusion avec les "s" long.
- Notre OCR Tesseract a également tendance à halluciner, c'est-à-dire à reconnaître des éléments qui n'existent pas.

Au vu de ces résultats, il serait intéressant de tester d'autres OCR. Une autre piste d'amélioration pour le futur serait d'explorer des techniques avancées de traitement d'image pour améliorer l'évaluation.

CONCLUSION GENERALE

Dans ce rapport, nous avons d'abord expérimenté et évalué divers outils d'OLR et d'OCR, tels que Tesseract, Layout Parser, Laypa, Grobid, et Surya. Ces expérimentations nous ont conduit à proposer une méthodologie basée sur Layout Parser, utilisant un modèle de Detectron2, qui s'est révélé performant pour analyser les pages du Trévoux.

Cette méthodologie, visant à affiner le modèle Detectron2, a montré des résultats prometteurs. Toutefois, les évaluations que nous avons menées suggèrent que l'amélioration de la quantité et de la qualité de notre base d'entraînement constitue la prochaine étape cruciale pour obtenir des résultats encore meilleurs.

BIBLIOGRAPHIE

- Karpinski, Romain, et Abdel Belaid. « Rapport Evaluation des OCR ». Report, LORIA - Université de Lorraine, 2016. https://inria.hal.science/hal-01356824.

- Shen, Zejiang, Ruochen Zhang, Melissa Dell, Benjamin Charles Germain Lee, Jacob Carlson, et Weining Li. « LayoutParser: A Unified Toolkit for Deep Learning Based Document Image Analysis ». arXiv.org, 29 mars 2021. https://arxiv.org/abs/2103.15348v2.

TABLE DES ILLUSTRATIONS

TABLE DES MATIERES