

Projet ACTES IA

Pipeline dATA

Documentation technico-fonctionnelle

01 octobre 2020



Table des matières

[1. Prérequis techniques et installation du projet 2](#_Toc52543979)

[1.1. Versions des librairies 2](#_Toc52543980)

[1.2. Import et installation du projet de Gitlab 2](#_Toc52543981)

[2. Documentation technique 2](#_Toc52543982)

[3. Pipeline d’ingestion des requêtes 3](#_Toc52543983)

[3.1. Etapes du pipeline 3](#_Toc52543984)

[3.2. Limitations actuelles du pipeline 4](#_Toc52543985)

[3.3. Statistiques sur le nombre d’actes 4](#_Toc52543986)

[3.4. Description des principaux modules 4](#_Toc52543987)

[4. Base de données PostgreSQL 5](#_Toc52543988)

[4.1. Table obtenue après la pipeline 5](#_Toc52543989)

[4.2. Table pour l’outil d’annotation 6](#_Toc52543990)

[4.3. Table contenant les annotations 6](#_Toc52543991)

# Prérequis techniques et installation du projet

A date, le pipeline est exécutable manuellement. La mise en production (automatisation de l’exécution) sera réalisée à la fin du projet après que la gestion des flux des données aura été définie avec le responsable technique. Dans cette section, nous décrivons les librairies utilisées et le processus d’installation du projet dans un nouvel environnement.

## Versions des librairies

L’ensemble des traitements est réalisé en **Python 3.7**. Les bibliothèques suivantes sont nécessaires à la bonne exécution du pipeline.

|  |  |  |
| --- | --- | --- |
| **Bibliotèque** | **Version** | **Fonction** |
| **nltk** | 3.4.5 | Bibliothèque de pretraitement du texte |
| **numpy** | 1.18.1 | Bibliothèque scientifique de calculs |
| **pandas** | 1.0.1 | Bibliothèque de gestion et transformation de DataFrames |
| **pdf2image** | 1.13.1 | Bibliothèque permettant de convertir les pages d'un PDF en images |
| **Pillow** | 7.0.0 | Bibliothèque de traitement d'images |
| **psutil** | 5.6.7 | Bibliothèque de monitoring du système |
| **py7zr** | 0.6 | Bibliothèque de decompression des fichiers 7z |
| **PyPDF2** | 1.26.0 | Bibliothèque d’extraction des métadonnées PDF |
| **pytesseract** | 0.3.3 | Bibliotèque d'océrisation d'images |
| **seaborn** | 0.10.0 | Bibliothèque d’affichage de graphiques |
| **sqlalchemy** | 1.3.16 | Bibliothèque pour requêter et écrire dans des bases de données SQL |
| **tika** | 1.23.1 | Bibliothèque qui permet d'extraite le texte brut des PDF |

## Import et installation du projet de Gitlab

* Le projet est disponible sur le Gitlab de Starclay à l’adresse <https://gitlab.starclay.fr/ms9/dgcl/actes-ia-data>
* L’accès à ce dépôt de code est donné par Joseph Assu Ondo (jassuondo@starclay.fr), le pilote du projet.
* Pour importer le projet sur un nouvel environnement, dans un dossier taper la commande git clone [git@gitlab.starclay.fr:ms9/dgcl/actes-ia-data.git](mailto:git@gitlab.starclay.fr:ms9/dgcl/actes-ia-data.git) après avoir configuré votre clé ssh sur Gitlab

# Documentation technique

Toute la documentation technique du projet est disponible sur Gitlab. Ci-après les éléments que vous y trouverez.

|  |  |  |
| --- | --- | --- |
| **Document** | **Description** | **Lien Gitlab** |
| ReadME.md | Document principal rappelant le fonctionnement du dépôt Gitlab et faisant référence aux autres documents | <https://gitlab.starclay.fr/ms9/dgcl/actes-ia-data/blob/develop/README.md> |
| docs/CONTRIBUTING.md | Décrit le processus pour contribuer (ajout de code) sur le projet | <https://gitlab.starclay.fr/ms9/dgcl/actes-ia-data/blob/develop/docs/CONTRIBUTING.md> |
| docs/README.md | Description du processus de génération de la documentation du code | <https://gitlab.starclay.fr/ms9/dgcl/actes-ia-data/blob/develop/docs/README.md> |
| docs/\_build/pdf/Python.pdf | Contient la description des éléments implémentés dans le code (fonctions, classes, modules) | <https://gitlab.starclay.fr/ms9/dgcl/actes-ia-data/blob/develop/docs/_build/pdf/Documentation%20Actes%20IA.pdf> |

# Pipeline d’ingestion des requêtes

## Etapes du pipeline

* **Extraction du texte des actes :**
  + **Répertoires contenant les actes :** Chaque répertoire contient les actes d’un département, les actes des départements 30, 44, 75, 77 se situent respectivement dans les répertoires *DONNEES\_20200504/DEPT\_30\_2018, DONNEES\_20200504/DEPT\_44\_2018, DONNEES\_20200504/DEPT\_75\_2018, DONNEES\_20200504/DEPT\_77\_2018*
  + **Traitement :**

Le pipeline va dans un premier temps récupérer la liste des fichiers pdf du dossier spécifié et le nombre de thread à utiliser (le nombre d’exécution parallèle à effectuer).

Ensuite, il lance en parallèle l’OCR des fichier pdf. L’océrisation est gérée avec la bibliothèque Apache Tika pour les PDF textes et avec Tesseract pour les pdf scannés. La distinction du type de PDF (scan ou texte) est réalisée grâce à PyPDF2.

* **Récupération des méta données associées aux requêtes** :
  + **Fichiers de méta données**: *DONNEES\_PROJET\_ACLIA/t\_documents\_dept\_X.csv* et *DONNEES\_PROJET\_ACLIA/t\_actes\_dept\_X.csv* où X représente le numéro du département
  + **Traitement** :

Le fichier t\_documents\_dept\_X.csv contient les informations de chaque acte relatif au document en lui-même (taille du fichier, extension du fichier, date de dernière modification)

Le fichier t\_actes\_dept\_X.csv contient les métadonnées de chaque acte (nature de l’acte, matière de l’acte, …)

On effectue une jointure des fichiers pour chaque département à l’aide du numéro de l’acte (colonne noacte pour le fichier t\_actes\_dept\_X.csv et t\_acte\_noacte pour le fichier t\_documents\_dept\_X.csv)

* **Jointures des requêtes aux méta données**
  + **Traitements :**

Les actes issus de l’océrisation sont joints aux méta données à l’aide du numéro de l’acte (colonne folder pour la table contenant les actes océrisés et noacte pour la table contenant les métadatas) et le résultat est stocké dans la table finale *dw\_actes*

* + Table de sortie : *dw\_actes*

## Limitations actuelles du pipeline

**L’océrisation des requêtes ne fonctionne pas sur les écritures manuscrites, les signatures, les tampons et les logos**. A la place du texte contenu dans ces éléments, il y aura des successions de caractères et de ponctuations non intelligibles. Ces cas représentent une partie non importante des océrisations.

Également notre moteur d’océrisation ne parvient pas à extraire le texte d’une très petite partie des Pdf (364). Nous les avons marqués « illisibles » et n’avons pas trouvé une solution à cette anomalie. Comme la volumétrie est faible, l’anomalie est mineure.

## Statistiques sur le nombre d’actes

Dans le tableau ci-dessous, les statistiques concernant les types de PDF à l’issue du pipeline sont présentées.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Type de PDF | **Dept 30** | **Dept 44** | **Dept 75** | **Dept 77** |  |
| **PDF Textes** | 10 147 | 61 410 | 9 891 | 20 002 | 101 450 |
| **PDF Images** | 43 432 | 73 054 | 6 153 | 42 708 | 165 347 |
| **Total** | 53 579 | 134 464 | 16 044 | 62 710 | 266 797 |

|  |  |
| --- | --- |
| Textes extraits | **Nb fichiers** |
| **PDF océrisés** | 266 433 |
| **PDF illisibles (\*)** | 364 |
| **Total** | 266 797 |

(\*) : nous avons marqué « illisibles » les Pdf que le moteur d’océrisation n’a pas réussi à lire.

## Description des principaux modules

Les principaux scripts du pipeline sont dans le dossier ‘pipeline’ :

* **bdd.py**

Description :permet de gérer l’envoi et la réception de tables à partir de la base de données PostgreSQL

* **extract.py**

Description :contient toutes les fonctions utilisées pour extraire le texte de fichiers pdf textes et celles pour extraire le texte par OCR des fichiers pdf scannés. Il contient également les fonctions permettant d’extraire les meta données des PDF.

* **pipeline.py**

Description :Classe permettant à partir d’un dossier contenant les fichiers pdf, d’extraire le texte des PDFs puis d’effectuer la jointure avec les métadonnées.

* **main.py**

Description :Utilise la classe Pipeline ci-dessus pour extraire le texte d’un PDF et le sauvegarder au format csv.

* **utils.py**

Description :contient les fonctions de gestion des fichiers ainsi que celles pour lancer l’exécution de l’OCR en parallèle (multithreading / multiprocessing).

Le script permettant de lancer le pipeline sur un dossier contenant des actes se situe dans le dossier ‘pipeline/scripts’ :

* **pipeline\_on\_data.py**

Description : Lance le pipeline à partir du dossier contenant les actes et du dossier contenant les métadonnées

-d <chemin\_actes> : Chemin vers le dossier contenant les actes à océriser

-m <chemin\_metadonnées> : Chemin vers le dossier contenant les métadonnées

Exemple : *python pipeline\_on\_data.py -d /documents-AR-actes\_DPT30\_20180109\_20180113/ -m /DONNEES\_PROJET\_ACLIA/*

# Base de données PostgreSQL

## Table obtenue après la pipeline

La base de données du projet est « dgcl ». Le tableau ci-dessous présente les principales colonnes obtenues après traitement des actes dans la pipeline. La table est présente dans la base de données sous le nom « dw\_actes »

|  |  |  |
| --- | --- | --- |
| **Colonne** | **Type** | **Description** |
| noacte | Chaine de caractères | Numéro unique à chaque acte |
| path | Chaine de caractères | Chemin du fichier océrisé |
| filename | Chaine de caractères | Nom du fichier |
| extracted\_text | Chaine de caractères | Texte issu de l’océrisation |
| pdf\_type | Chaine de caractères | “image” si c’est un pdf scanné, “text” sinon |
| num\_pages | Nombre | Nombre de pages du fichier |
| num\_files | Nombre | Nombre de fichiers comprenant l’acte et ses pièces jointes |
| pj\_acte\_principal | Chaine de caractères | Reference au filename de l’acte quand c’est une pièce jointe (null sinon) |
| objetacte | Chaine de caractères | Correspond à l’objet renseigné lors de l’envoi de l’acte |
| soumistransmission | Binaire | « 1 » si l’acte est soumis à la transmission, « 0 » sinon |
| t\_matiere\_codematiere | Chaine de caractères | Tous les niveaux de matière renseignés |
| t\_natureacte\_codenature | Nombre | Niveau de l’acte |
| t\_site\_nodepartement | Nombre | Numéro du département de l’acte |

## Table pour l’outil d’annotation

Pour permettre de faire fonctionner l’outil d’annotation, une table avec les colonnes ci-dessous est nécessaire. Elle est présente dans la base de données sous le nom « test\_annotations »

|  |  |  |
| --- | --- | --- |
| **Colonne** | **Type** | **Description** |
| noacte | Chaine de caractères | Numéro unique à chaque acte |
| filename | Chaine de caractères | Nom du fichier |
| nature | Nombre | Nature de l’acte |
| matiere | Chaine de caractères | Tous les niveaux de matière renseignés |
| objetacte | Chaine de caractères | Correspond à l’objet renseigné lors de l’envoi de l’acte |
| texte | Chaine de caractères | Texte issu de l’océrisation |
| pj\_acte\_principal | Chaine de caractères | Reference au filename de l’acte quand c’est une pièce jointe (null sinon) |
| dept | Nombre | Numéro du departement |
| iddocument | Nombre | Id du document |
| is\_recrutement | Booléen | True si l’objet contient le mot « recrutement » |
| is\_avenant | Booléen | True si l’objet contient le mot « avenant » |
| statue | Booléen | True si l’acte est annoté |
| step | Nombre | Étape d’annotation du document |

## Table contenant les annotations

Les annotations réalisées sur l’outil sont stockées dans une nouvelle table « app\_annotation ».

|  |  |  |
| --- | --- | --- |
| **Colonne** | **Type** | **Description** |
| username | Chaine de caractères | Identifiant de l’utilisateur |
| date | Date | Date à laquelle l’acte a été annoté |
| qcm | Json | Contient l’ensemble des annotations relatives à un acte donné |
| iddocument | Nombre | Id du document |
| filename | Chaine de caractères | Nom du fichier |