Documentation mandat

Table des matières

[1. Contexte du Mandat 1](#_Toc155726335)

[2. Résultats 1](#_Toc155726336)

[3. Informations sur l'exécution des tâches et résultats obtenus 2](#_Toc155726337)

[Reconstituions des données issues du projet de 2022 2](#_Toc155726338)

[Scripts créés explications 2](#_Toc155726339)

[ProductionSets 2](#_Toc155726340)

[IterationModels 2](#_Toc155726341)

[ProductionCSV\_1829 / ProductionCSV\_1837 / ProductionCSV\_1846 / ProductionCSV\_1850 2](#_Toc155726342)

[Installation du server Escriptorium sur le HP 3](#_Toc155726343)

[GPG key pour pass 3](#_Toc155726344)

[Pass 3](#_Toc155726345)

[Compte docker hub 3](#_Toc155726346)

[Identifiants pour se connecter sur le server Escriptorium 3](#_Toc155726347)

[Utilisation du server 3](#_Toc155726348)

[4. Réflexions apparues lors du mandat et poursuite du projet 4](#_Toc155726349)

[Acquisition du matériel 4](#_Toc155726350)

[Installation d’un server Escriptorium 4](#_Toc155726351)

[Installation de docker et git : 4](#_Toc155726352)

[Installation de Kraken 5](#_Toc155726353)

[Première étape : installation de pyenv 5](#_Toc155726354)

[Deuxième étape : création d’un environnement virtuel et installation de Kraken 5](#_Toc155726355)

# Contexte du Mandat

En automne 2022, l'AEV et la VALAIS-WALLIS TIME MACHINE ont mandaté Thibault Clérice pour développer un modèle de reconnaissance d'écriture manuscrite sur les recensements valaisans. Les transcriptions destinées à entraîner le modèle ont été réalisées par Darius Schlaeppi, Clémence Rudaz et moi-même, sur eScriptorium. Les recensements de 1870 et de 1880 ont été traités, en allemand et en français. En automne 2023, il a été décidé de poursuivre ce projet.

# Résultats

Lors de ce mandat, j’ai effectué les actions suivantes :

* Installation d’un server sur le HP-Linux pour utiliser E-scriptorium
* Installation de Kraken sur le HP-Linux
* Création de scripts pour obtenir les résultats en CSV
* Création de 4 fichiers CSV pour les années 1829, 1839, 1846, 1850

Le server EScriptorium n’a pas pu être utilisé, car l'ordinateur ne disposait pas de la puissance nécessaire pour entraîner un modèle. Cependant, il a été tout à fait possible de segmenter et transcrire des pages des recensements.

L’installation de Kraken a permis l’itération du modèle de segmentation « blla.mlmodel» et du modèle «KrakenTranscription.mlmodel » pour la transcription sur les recensements de 1829, 1839, 1846 et 1850.

# Informations sur l'exécution des tâches et résultats obtenus

## Reconstituions des données issues du projet de 2022

En 2022, Thibault Clérice a dirigé le projet de reconnaissance d'écriture. Les données générées par les stagiaires, ainsi que les modèles créés, sont disponibles sur GitHub à l'adresse suivante : <https://github.com/PonteIneptique/valais-recensement/blob/main/.github/workflows/htr-united-workflow.yml>. De plus, Une documentation sur les règles de transcriptions appliquée en 2022 a été créer. Toutes les fichiers, modèles et informations liés au mandat de Thibault ont été regroupés dans un dossier sur le HP-Linux intitulé « ThibaultClerice\_ECR\_2022 ».

## Scripts créés explications

### ProductionSets

Ce script a pour fonction de créer des ensembles de 25 images. Cette étape s'est révélée nécessaire pour le bon fonctionnement du script IterationModel.

### IterationModels

Afin d'automatiser l'itération des modèles de transcription et de segmentation avec Kraken, j'ai développé un script Python. Ce script permet de parcourir X ensembles de 25 images, assurant ainsi une répartition équilibrée de l'itération des modèles. Cette approche vise à éviter une automatisation trop autonome qui pourrait négliger des problèmes potentiels, tout en évitant une automatisation peu autonome. Le script génère des métadonnées au format Alto.

Pour évaluer le temps nécessaire à la création des métadonnées, j'ai comptabilisé les heures pendant lesquelles le script "IterationModels" a été en cours d'exécution.

1837 : 53 heures pour 4662 fichiers jpg / 8.20 GO

1846 : 67 heures pour 2952 fichiers jpg/ 37.6 GO

1829 : 92 heures pour 1816 fichiers jpg / 39.9 GO

1850 : pour 2303 fichiers jpg / 16.6 GO

### ProductionCSV\_1829 / ProductionCSV\_1837 / ProductionCSV\_1846 / ProductionCSV\_1850

À partir des métadonnées générées par les modèles de transcription et de segmentation, un fichier CSV par année a été créé pour être utilisé dans le moteur de recherche en ligne. Chaque fichier CSV est organisé en fonction du nom du fichier XML issu de la numérisation, suivi de l'ordonnancement de haut en bas selon la position y des données. Pour les années 1829 et 1846, l'emplacement sur la page gauche ou droite est également indiqué.

En cas d'absence de nom de famille pour un prénom (ce qui peut se produire si l'OCR n'a pas reconnu les guillemets anglais, ou si tout simplement le nom de famille n’était pas mentionné), il est remplacé par un "?". Cela suggère que le nom de famille précédent correspond probablement à ce prénom.

## Installation du server Escriptorium sur le HP

Un serveur Escriptorium a été installé sur le HP-Linux dans l'intention de segmenter et transcrire des pages de recensement en vue d'améliorer les modèles de Thibault Clérice. Malheureusement, cette démarche n'a pas pu être concrétisée en raison de la limitation de performances matérielles, qui ne permettait pas d'entraîner des modèles d'OCR de manière efficace. Les détails concernant les éléments nécessaires à l'installation du serveur sont présentés ci-dessous.

### GPG key pour pass

Nom réel : R3nd0mN4m3

Adresse E-mail : [marie-caroline.schmied@admin.vs.ch](mailto:Marie-Caroline.SCHMIED@admin.vs.ch)

Mot de passe/phrase secrète : 3lepassemot

### Pass

Utilisateur : Delphine

Mot de passe : 5ChatsBleus

### Compte docker hub

* Adresse E-mail : Delphine.MAMIE@admin.vs.ch
* delphine710
* Mot de passe : le mot de passe710

### Identifiants pour se connecter sur le server Escriptorium

* DJANGO\_SU\_NAME= adminDelphine
* DJANGO\_SU\_EMAIL = [Delphine.MAMIE@admin.vs.ch](mailto:Delphine.MAMIE@admin.vs.ch)
* DJANGO\_SU\_PASSWORD = lemotdepasseDelphine

## Utilisation du server

Entrer ces lignes dans le Terminal :

$ cd escriptorium

$ sudo systemctl start docker

$ docker compose pull

$ docker compose up -d

Puis cliquer sur ce lien :

<http://localhost:8080/>

Lorsque le travail est fini, il faut stopper le server avec cette ligne :

$ sudo systemctl stop docker

# Réflexions apparues lors du mandat et poursuite du projet

## Acquisition du matériel

Si la poursuite du projet implique l'entraînement de nouveaux modèles de transcription et de segmentation, il serait impératif d'acquérir du matériel informatique plus puissant. Thibault Clérice avait établi des devis correspondant aux besoins nécessaires.

## Installation d’un server Escriptorium

Cette marche à suivre détaille le processus d'installation d'un serveur Scriptorium, en cas d'acquisition de nouveau matériel et de nécessité d'une nouvelle installation.

Pour installer un server, j’ai suivi les instructions se trouvant ici : <https://gitlab.com/scripta/escriptorium/-/wikis/docker-install>

### Installation de docker et git :

* Installation Git
* Création une GPG key pour pass
* Installalation docker(<https://docs.docker.com/engine/install/ubuntu/>)[[1]](#footnote-1)
* Actuellement Docker version 20.10.21, build 20.10.21-0ubuntu~18.04.3
* docker’s post-installations : <https://docs.docker.com/engine/install/linux-postinstall/>
* Configure default logging driver : <https://docs.docker.com/config/containers/logging/json-file/>
* Installation de Install nvidia-docker2 de cette manière : <https://docs.nvidia.com/datacenter/cloud-native/container-toolkit/latest/install-guide.html> « with APT »

Comme mentionné précédemment, le serveur Escriptorium n'a pas pu être utilisé en raison d'une puissance insuffisante de l'ordinateur pour entraîner des modèles de transcription ou de segmentation. Pour évaluer la possibilité d'entraîner de tels modèles, j'ai segmenté et transcrit une vingtaine de pages de 1837. Si la décision est prise de poursuivre l'entraînement des modèles à partir des recensements, voici les règles que j'ai suivies pour la segmentation et la transcription :

En ce qui concerne la transcription, j'ai suivi les règles appliquées pour les recensements de 1870 et 1880, comme indiqué dans le document 'documentation transcription'. Pour la segmentation, j'ai conservé les mêmes catégories que celles utilisées dans les recensements de 1870 et 1880 (details, header, col, signature). En ce qui concerne les modèles, j'ai utilisé « blla.mlmodel » pour la segmentation et « KrakenTranscription.mlmodel ». Pour la segmentation, le modèle n'est bien sûr pas adapté aux colonnes, mais il est parfaitement adapté à la sélection des lignes. Le modèle de transcription est très performant, avec quelques erreurs persistantes.

## Installation de Kraken

Après plusieurs tentatives et diverses erreurs, il a été déterminé que l'utilisation de Python 3.8.18 était nécessaire. Étant donné que le HP héberge plusieurs projets à partir d'une autre version de Python, Kraken a été installé dans un environnement virtuel.

### Première étape : installation de pyenv

Pyenv est un outil qui facilite le passage d'une version à une autre de Python pour différents projets. Pour ce faire, j'ai suivi ce tutoriel : <https://www.youtube.com/watch?v=3my06DUnApM&t=1s>. Voici les lignes de code qui ont été saisies dans le terminal :

1. $ curl https://pyenv.run | bash[[2]](#footnote-2)
2. $ sudo apt-get update
3. $ sudo apt-get install -y build-essential libssl-dev zlib1g-dev libbz2-dev \libreadline-dev libsqlite3-dev wget curl llvm libncurses5-dev libncursesw5-dev \xz-utils tk-dev libffi-dev liblzma-dev python3-openssl git[[3]](#footnote-3)
4. $ sudo apt-get update
5. $ pyenv install 3.8
6. $ sudo apt-get update
7. $ sudo apt install -y tk-dev[[4]](#footnote-4)
8. $ pyenv versions[[5]](#footnote-5)
9. $ cd “dossier/duprojet”
10. $ pyenv local 3.8[[6]](#footnote-6)

### Deuxième étape : création d’un environnement virtuel et installation de Kraken

1. $ python3 -m venv myenv
2. $ source myenv/bin/activate
3. $ pip install –upgrade pip
4. $ pip install kraken
5. $ kraken --version[[7]](#footnote-7)

1. J’ai spécifiquement suivi l’installation de Install Docker Desktop on Linux ici : <https://docs.docker.com/desktop/install/linux-install/> + <https://docs.docker.com/desktop/install/ubuntu/> [↑](#footnote-ref-1)
2. Lors de l'installation, une modification a été apportée dans le fichier bash. [↑](#footnote-ref-2)
3. Cette étape a été réalisée lors de l’installation de Python 3.12, ce qui a provoqué des problèmes de dépendances avec PyTorch. C’est pour cette raison que j’ai finalement opté pour Python 3.8 (https://pytorch.org/get-started/locally/#windows-prerequisites). Il est possible que cette étape ne soit pas nécessaire pour l’installation de Python 3.8. [↑](#footnote-ref-3)
4. Lors de l’installation de python 3.8, une erreur (qui semble être courante sur linux cf [tutoriel](https://www.youtube.com/watch?v=3my06DUnApM&t=1s)) est apparue relative à tk. Cette étape peut ne pas être nécessaire. [↑](#footnote-ref-4)
5. Cette commande permet de contrôler que la version Python 3.8 a bien été installée. [↑](#footnote-ref-5)
6. Argument «local» permet d’appliquer l’utilisation de python 3.8 uniquement où la ligne de commande est effectuée (cf. ligne9) [↑](#footnote-ref-6)
7. La version de kraken que j’ai utilisée est la 4.3.13 [↑](#footnote-ref-7)