

Kévin DESPOULAINS

Gaël GENDRON

Corentin GUILLOUX

Valentin FOUCHER

Enzo CRANCE

Charlotte RICHARD

Laure DU MESNILDOT

Timothée NEITTHOFFER

Elèves ingénieurs de l'INSA Rennes

Année universitaire 2018/2019

# Rapport de pré-étude Projet 4INFO

## Logiciel de génération de données d'apprentissage pour la reconnaiss- sance d'écriture manuscrite

### Encadrants

Bertrand COÜASNON (*IRISA*)Erwan FOUCHÉ & Julien BOUVET (*Sopra Steria*)

### Projet en collaboration avec

Jean-Yves LE CLERC (*Archives départementales*)Sophie TARDIVEL (*Doptim*)

# Table des matières

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Vocabulaire utile</b>	<b>2</b>
<b>3</b>	<b>Contexte du projet</b>	<b>3</b>
3.1	L'équipe de recherche IntuiDoc . . . . .	3
3.2	Doptim . . . . .	3
3.3	Les archives départementales d'Ille-et-Vilaine . . . . .	4
<b>4</b>	<b>Cahier des charges</b>	<b>5</b>
4.1	Les objectifs . . . . .	5
4.2	Description fonctionnelle des besoins . . . . .	8
<b>5</b>	<b>État de l'art</b>	<b>10</b>
5.1	Techniques de reconnaissance d'écriture . . . . .	10
5.2	Détecteur de lignes . . . . .	12
5.3	Format de description d'image . . . . .	15
5.4	Base de données . . . . .	15
5.5	Interface Homme-Machine . . . . .	16
5.6	Traitements d'images . . . . .	16
<b>6</b>	<b>Organisation du projet</b>	<b>17</b>
6.1	Méthodes de travail . . . . .	17
6.2	Répartition des tâches . . . . .	17
6.3	Estimation de la planification des tâches . . . . .	19
<b>7</b>	<b>Conclusion</b>	<b>20</b>
<b>8</b>	<b>Annexes</b>	<b>22</b>
8.1	Annexe A : Schémas résumant le projet dans chacun de ses modes de fonctionnement . . . . .	22

# Chapitre 1

## Introduction

Ce projet nous a été proposé par l'équipe [IntuiDoc](#) de l'[IRISA](#), en étroite collaboration avec la startup [Doptim](#) et avec le soutien de Jean-Yves LE CLERC, conservateur du patrimoine aux [archives départementales](#) d'Ille-et-Vilaine. Tout au long de l'année, nous serons encadrés par Bertrand COÜASNON, enseignant-chercheur membre d'IntuiDoc, Erwan FOUCHÉ, chef de projet chez [Sopra Steria](#), Julien BOUVET, ingénieur chez Sopra Steria également. Nous serons aussi accompagnés par Sophie TARDIVEL, responsable et *data scientist* chez Doptim.

Ce projet a pour but de fournir un programme permettant de concevoir des bases d'apprentissage automatiquement pour l'entraînement de divers systèmes de reconnaissance d'écriture manuscrite ainsi que leur exploitation. Ces reconnaiseurs seront, par exemple, capables de retranscrire de manière informatique des documents manuscrits (registres paroissiaux, registres d'état civil, documents d'entreprise, ...) pour les rendre plus exploitables. Ce projet permettra donc de gagner du temps sur la compréhension de documents anciens en rendant l'entraînement de systèmes complexes plus simple.

Le [chapitre 3](#) de ce rapport détaillera dans un premier temps le contexte du projet. Dans le [chapitre 4](#), nous expliciterons les différentes tâches qui doivent être réalisées, l'objectif mentionné dans le paragraphe précédent restant le principal. Nous étudierons ensuite dans le [chapitre 5](#) les différents outils à notre disposition. Enfin, nous conclurons sur l'organisation prévisionnelle du projet dans le [chapitre 6](#).

# Chapitre 2

## Vocabulaire utile

Avant de commencer, il est important de comprendre certaines notions utiles :

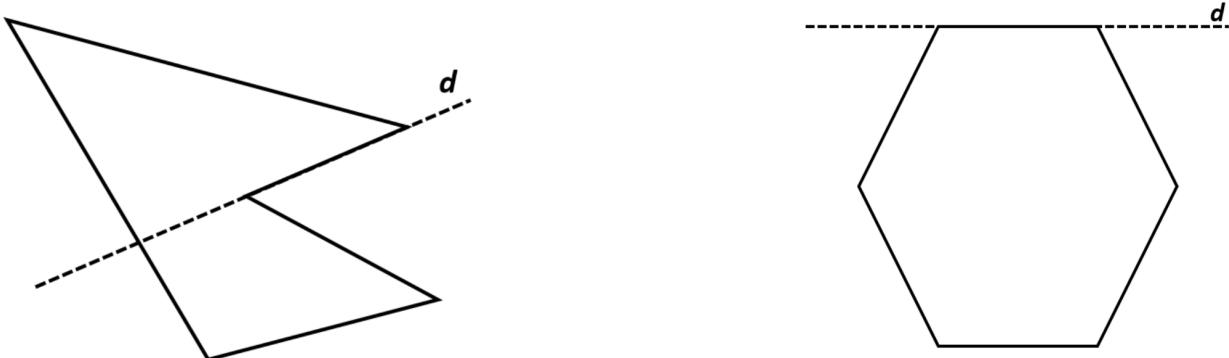
Le **deep learning** est une méthode d'apprentissage automatique basée sur une modélisation de type neuronal. Ainsi, un algorithme utilisant le *deep learning* apprend par lui-même et devient de plus en plus performant au fur et à mesure qu'il accumule les exemples. Il suffit de lui spécifier les paramètres du problème qu'il doit résoudre et de lui donner des exemples sur lesquels s'entraîner.

Une **base d'apprentissage** (ou base d'entraînement) est un ensemble d'exemples que l'on fournit à un algorithme de *deep learning* afin que celui-ci puisse apprendre sur ces exemples.

La **vérité terrain** est, dans le cas de notre projet, un ensemble de documents manuscrits annotés de diverses informations telles que la position des paragraphes dans les documents ou encore la transcription informatique des textes. Cette vérité a été établie au préalable par des humains et non de manière automatique.

Un **polygone concave** est un polygone complexe dont l'intérieur est un ensemble concave. D'une manière géométrique, on dit qu'un polygone est concave si lorsque l'on trace une droite passant par l'un de ses côtés, alors le polygone se retrouve en partie coupé par cette droite. Un polygone non concave est dit convexe.

Figure 1 : Exemples de polygones concave (à gauche) et convexe (à droite)



Une **imagette** est une image réduite d'une autre image. Dans le cas de notre projet, cela correspond à une partie de texte manuscrit découpée à partir du document initial.

Nous appellerons **retranscription** la transcription informatique d'un texte manuscrit.

# Chapitre 3

## Contexte du projet

Comme énoncé dans l'introduction, ce projet est en collaboration avec l'équipe IntuiDoc de l'IRISA, l'entreprise Doptim et les archives départementales d'Ille-et-Vilaine. Le produit de notre travail leur sera donc mis à disposition, directement ou indirectement, afin qu'ils puissent l'exploiter.

### 3.1 L'équipe de recherche IntuiDoc

Dans le cadre de ses recherches, l'équipe IntuiDoc de l'IRISA cherche à faire avancer le domaine de la reconnaissance d'écriture manuscrite afin de rendre plus accessibles des textes anciens qui sont souvent peu compréhensibles. Elle est en collaboration avec les archives départementales d'Ille-et-Vilaine. Il n'est pas simple d'écrire un programme qui reconnaît les textes manuscrits, c'est pourquoi la plupart des systèmes de reconnaissance d'écriture sont basés sur des algorithmes intelligents. Ces algorithmes sont souvent formés de réseaux de neurones qui ont besoin d'apprendre à reconnaître les différents caractères, quels que soient la langue et le style du rédacteur. Pour apprendre, ils ont besoin d'un grand nombre d'exemples (plusieurs milliers) qui sont longs à construire à la main.

Dans ce contexte, les exemples d'apprentissage (appelés base d'apprentissage dans la suite de ce rapport) sont des associations entre les textes manuscrits et leurs retranscriptions informatiques. Ainsi, l'algorithme apprend à reconnaître les caractères en comparant sa sortie avec la retranscription fournie. Notre projet, qui a été proposé par l'équipe IntuiDoc, est donc de construire un système qui permet de générer des bases d'apprentissage à partir d'images et d'une vérité terrain de manière automatique.

### 3.2 Doptim

Doptim est une entreprise spécialisée dans le *Big Data* et l'analyse de données. Elle a été fondée par Sophie TARDIVEL, qui sera notre contact dans l'entreprise. Son but premier est de créer une communauté de *data scientists* et de *data engineers* qui auraient l'ambition d'optimiser et de maîtriser la gestion des données. Doptim est aujourd'hui investie dans un projet de service en ligne permettant aux généalogistes de gagner du temps dans la fouille et le décryptage de documents manuscrits numérisés. De plus, Doptim est en collaboration avec l'équipe de recherche IntuiDoc. Cette collaboration a pour but de créer un système de reconnaissance d'écriture manuscrite qui aidera les archivistes.

---

### 3.3 Les archives départementales d'Ille-et-Vilaine

Les archives départementales d'Ille-et-Vilaine, situées à Rennes, regroupent plusieurs millions de documents manuscrits datant, pour les plus anciens, du début du XI<sup>ème</sup> siècle. Ces documents n'y sont pas seulement entreposés mais aussi numérisés pour que n'importe qui puisse les consulter, soit en version physique aux archives, soit en version numérique depuis n'importe où. Il existe actuellement des moteurs de recherche permettant de retrouver les documents grâce à des annotations ou des mots-clés, mais leur utilisation reste limitée car les documents sont souvent peu lisibles. La collaboration entre l'équipe IntuiDoc et Doptim a pour but de retranscrire la totalité du contenu d'un document manuscrit sous forme de texte numérique pour que la recherche soit plus efficace et la consultation plus agréable. Cet outil pourra facilement être intégré à la chaîne d'archivage des documents puisqu'il pourra être exécuté juste après la numérisation, sans intervention humaine.

L'outil final de notre projet qui a pour but d'être compatible avec plusieurs types de reconnaiseurs d'écriture manuscrite sera pas notre propriété, mais nous le rendrons public et utilisable par tout le monde. Ce sera un logiciel libre. La collaboration que nous avons avec l'IRISA, les archives départementales d'Ille-et-Vilaine et Doptim nous permettent d'orienter notre projet afin de nous faire une idée de ce qu'un utilisateur pourrait en attendre.

# Chapitre 4

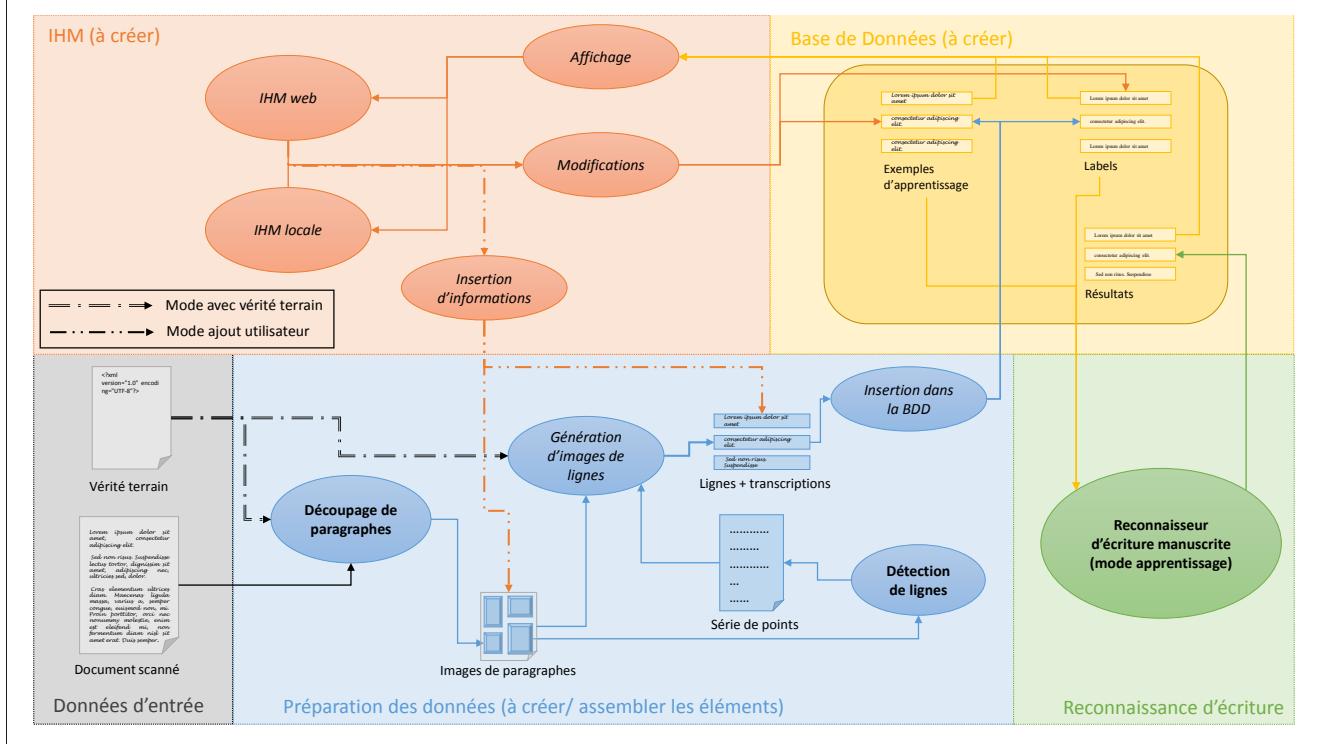
## Cahier des charges

Dans cette partie, nous allons détailler les différentes caractéristiques de notre projet. Nous commencerons par étudier les différents objectifs du projet, puis nous en ferons une description fonctionnelle.

### 4.1 Les objectifs

L'objectif premier du projet est de créer un logiciel de génération de bases d'apprentissage pour un système de reconnaissance d'écriture manuscrite. Néanmoins, le projet peut être divisé en sous-parties illustrées par le schéma de la figure 2. Celui-ci, bien que complexe, peut aider le lecteur à comprendre les liens entre tous les objectifs. Les éléments en **gras** sur le schéma nous sont fournis et sont à intégrer dans le projet.

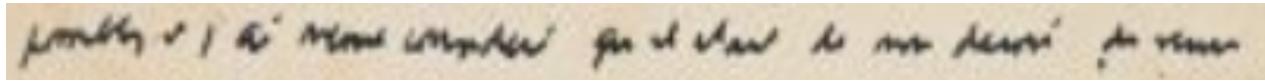
Figure 2 : Schéma résumant les différents aspects du projet



#### 4.1.1 Traitement des documents en entrée

Afin de générer des bases d'apprentissage, notre programme prendra en entrée des documents manuscrits numérisés et leurs retranscriptions. On qualifie les retranscriptions de vérité terrain car elles ont été réalisées par des humains et sont donc censées être justes. Le premier objectif est donc de découper correctement les documents manuscrits sous la forme d'imagettes. Ces imagettes peuvent correspondre à des lignes ou à des paragraphes des documents initiaux.

**Figure 3 : Exemple d'imagette**



Le découpage en lignes se fait à l'aide de plusieurs outils : deux outils de détection de lignes qui nous sont fournis, et un outil de découpage de paragraphes qui sera à développer au cours du projet. Cette partie correspond à la zone du schéma sur fond bleu. Afin de générer les imagettes, nous avons deux possibilités :

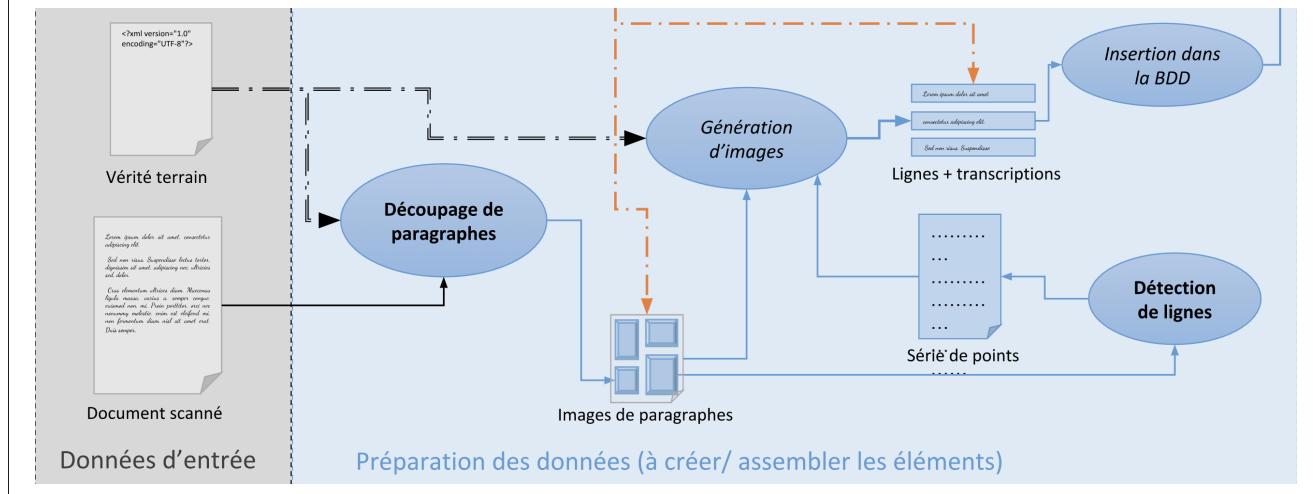
- option détection sur le document (figure 4) : on détecte la position des lignes sur l'entièreté du document, on découpe en paragraphes puis on relie les lignes détectées aux paragraphes ;
- option détection sur les paragraphes (figure 5) : on découpe d'abord les paragraphes, puis on détecte la position des lignes dans chacun des paragraphes.

Ces deux possibilités ont des avantages et des inconvénients. La première possibilité a l'avantage d'être plus précise. En effet, sur un document complet, on peut utiliser un meilleur outil de détection de lignes (à base de réseaux de neurones à convolution) qui analyse la forme globale du document. L'inconvénient est qu'il faut relier les lignes détectées à des paragraphes qui n'ont pas forcément une forme rectangulaire mais souvent plus complexe. Il faudra donc utiliser des outils géométriques pour détecter l'appartenance d'un point à un polygone qui peut être concave.

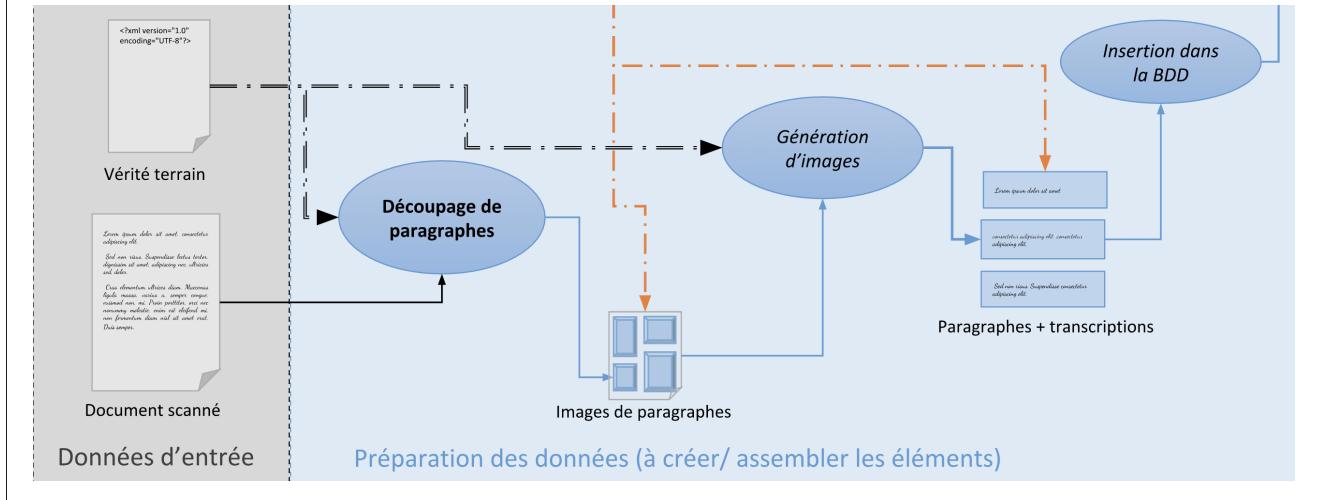
La seconde possibilité est moins précise car l'outil de détection de lignes précédent ne peut pas s'appliquer à un simple paragraphe. Nous devrons donc utiliser un autre outil (à base de floutage) qui peut traiter n'importe quel format de document mais moins en détail. L'avantage de cette possibilité est que nous n'aurons pas à relier les lignes aux paragraphes puisqu'elles y seront directement détectées. Nous avons cependant choisi d'implémenter les deux possibilités pour rendre notre système compatible avec une plus grande variété de documents.

Le schéma représentant les objectifs du projet montre un mode de fonctionnement (partie en bleu) où chaque imagette contient une seule ligne du document. Mais le projet doit contenir un second mode où les imagettes représentent des paragraphes et où la partie détection de ligne est absente. Ce second mode de fonctionnement permettra de générer des bases d'apprentissage pour des systèmes travaillant sur des paragraphes entiers et non simplement sur des lignes.

**Figure 4 : Partie traitement des données du projet (avec détection de lignes)**



**Figure 5 : Partie traitement des données du projet (sans détection de lignes)**



Dans un premier temps, nous travaillerons sur des documents provenant de la base **Maurdor**. Cette base, contenant des documents manuscrits numérisés ainsi que leur retranscription, présente l'avantage d'avoir, en plus, les informations sur la position des paragraphes et des retours à la ligne dans le document qui permettront de déduire la position des lignes (tout cela dans un format spécifique dérivé du XML, le [format GEDI](#)). Nous utiliserons ces informations pour découper les paragraphes. Par la suite, notre système devra être capable de prendre en entrée un autre format de documents, le format PiFF<sup>[6]</sup>, qui contient également les données de position des paragraphes dans une image. Nous expliquerons pourquoi nous avons choisi ce format par la suite.

#### 4.1.2 Mise en relation des imagettes et de leur retranscription

Afin que les imagettes puissent être exploitable, il faut les mettre en relation avec leur retranscription numérique (contenue dans la vérité terrain). Cet objectif est réalisable grâce aux données de position qui accompagnent les retranscriptions des fichiers d'entrée. Ces couples imagette-retranscription seront ensuite enregistrés dans une base de données qui constituera la base d'apprentissage d'un système de reconnaissance d'écriture manuscrite. Cette partie correspond également à la partie en bleu du schéma.

#### 4.1.3 Création d'une base de données

La base de données contient donc les couples imagette-retranscription ainsi formés (partie jaune du schéma). Il pourrait être intéressant de stocker, en plus, un texte généré par un reconnaissleur d'écriture pour pouvoir tout conserver dans la même base.

La plupart des systèmes de *deep learning* fonctionnent en trois modes : apprentissage, évaluation, utilisation. Le système de reconnaissance d'écriture manuscrite est situé dans la partie verte du schéma. La base de données que nous allons développer devra permettre de manipuler trois jeux de données différents :

- tout d'abord, en mode d'apprentissage, notre logiciel extrait de la base les deux premiers jeux de données (les imagettes et leurs retranscriptions associées) et les fournit au système de reconnaissance pour qu'il puisse s'entraîner ;
- ensuite, l'utilisateur doit pouvoir choisir un autre jeu de données pour évaluer son reconnaissleur ; cela permet de vérifier la qualité du reconnaissleur après un apprentissage ; ainsi, si l'on utilise une seule base pour stocker les ensembles d'apprentissage et d'évaluation, il pourrait être intéressant d'étiqueter les données pour indiquer dans quelle situation elles seront utilisées ;
- enfin, en mode de production, le reconnaissleur est supposé être bien entraîné ; il n'a que les imagettes en entrée et il fournit une retranscription numérique (un label) pour des documents sans vérité terrain ; il permet alors d'aider l'utilisateur à retranscrire numériquement un document manuscrit scanné ; ce label généré pourrait donc être intégré à la base de données comme nouvelle vérité terrain si l'utilisateur veut perfectionner son reconnaissleur.

---

Chaque mode de fonctionnement est illustré par un schéma en annexe A.

#### 4.1.4 Création d'une IHM permettant la modification de la base de données

Il est possible que la vérité terrain ne soit pas toujours fournie en entrée ou qu'elle soit incorrecte. Nous devons donc développer un logiciel permettant à l'utilisateur de rentrer lui-même une retranscription numérique et de donner la position des paragraphes (mode ajout utilisateur sur le schéma). On doit également pouvoir supprimer des couples imagette-retranscription de la base de données si jamais l'utilisateur considère qu'elles n'apporteront rien de bénéfique à l'apprentissage du reconnaiseur. Les données qu'on pourrait vouloir supprimer seraient des données trop bruitées, trop dégradées, mal coupées, etc. Il pourrait cependant être intéressant de les conserver pour que le reconnaiseur apprenne à reconnaître des textes abîmés. Cet objectif est représenté dans la partie rouge du schéma (IHM locale) qui doit pouvoir accéder à la base de données et la modifier.

#### 4.1.5 Utiliser un format spécifique et convertible en sortie

Notre logiciel générera des couples imagette-retranscription dans un format spécifique qui pourra ensuite être facilement modifié, cela dans le but d'être compatible avec les formats d'entrée de plusieurs systèmes de reconnaissance d'écriture manuscrite sans trop d'efforts. Plusieurs choix s'offrent à nous. Le format des données en entrée de la base Maurdor est GEDI. Cependant, les formats d'autres bases sont différents. GEDI n'est donc pas un format standard. Le format PiFF est un format de description d'images, mis en place par l'équipe IntuiDoc et par d'autres laboratoires, qui se veut générique et facilement convertible dans d'autres formats. Il nous a donc paru raisonnable de choisir d'utiliser ce format dans notre projet. Nous aurons donc un convertisseur du format d'entrée GEDI en PiFF et un convertisseur de PiFF vers différents formats de librairies de *deep learning*.

#### 4.1.6 Création d'une seconde IHM permettant d'observer les imagettes et les résultats d'un reconnaiseur

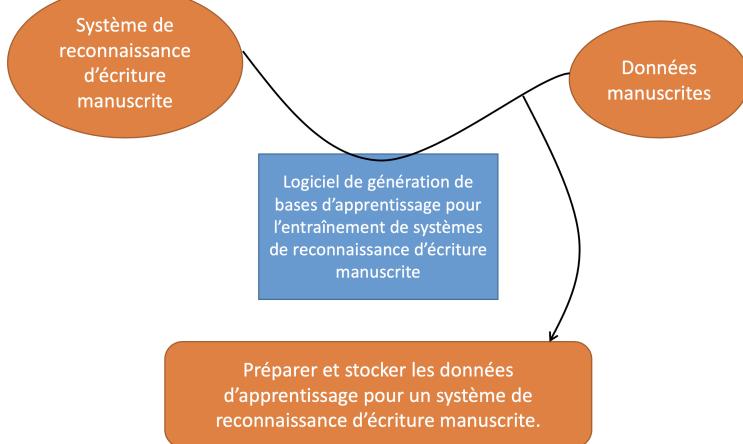
Pour vérifier le bon fonctionnement de notre programme de génération de bases d'apprentissage, nous avons à notre disposition un reconnaiseur d'écriture manuscrite nommé Laia[8] dont nous parlerons plus tard. Pour voir si notre système est efficace, il pourrait être intéressant de créer une seconde interface permettant d'observer les imagettes et les résultats du reconnaiseur. Cet objectif est optionnel car il sort légèrement du travail qui nous est demandé mais il sera utile pour faire une démonstration du reconnaiseur lors de la présentation finale de notre projet. On pourra ainsi montrer ce que notre travail aura pu apporter à la reconnaissance d'écriture manuscrite. Ce point ne sera probablement pas long à réaliser car il peut simplement consister en une amélioration de l'IHM précédente. Pour rendre la démonstration plus ludique, on pourrait aussi créer une IHM sous forme de jeu entre une personne et le reconnaiseur. Par exemple, la personne et le reconnaiseur pourraient avoir à retranscrire un texte manuscrit le plus vite possible. Cet objectif est représenté dans la partie rouge du schéma (IHM web) qui doit comme la première interface accéder à la base de données et la modifier.

### 4.2 Description fonctionnelle des besoins

#### 4.2.1 Diagramme bête à cornes

Le diagramme de la figure 6 permet d'exprimer le besoin lié au projet. Ainsi, notre logiciel de traitement de données manuscrites aura pour but principal de préparer et stocker des données d'apprentissage pour des systèmes de reconnaissance d'écriture manuscrite.

**Figure 6 : Diagramme bête à cornes**

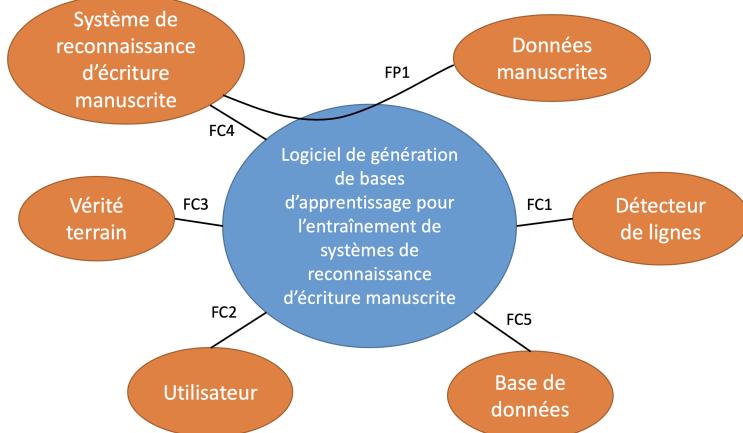


#### 4.2.2 Diagramme pieuvre

Le diagramme pieuvre de la figure 7, quant à lui, permet d'exprimer les différentes fonctions contraintes qui découlent de l'objectif principal du projet. De ce fait, notre logiciel devra interagir avec différents acteurs afin de s'intégrer parfaitement dans l'optique du projet :

- **FP1** : tout d'abord, le premier objectif (et l'objectif principal du projet) est de préparer et stocker les données d'apprentissage pour un système de reconnaissance d'écriture manuscrite ;
- **FC1** : ensuite, il devra utiliser le détecteur de lignes fourni afin de découper les phrases des textes d'entrée en imagettes ;
- **FC2** : il devra également permettre à l'utilisateur d'interagir avec les données traitées afin de les vérifier et/ou de les modifier si nécessaire, le tout par le biais d'une Interface Homme-Machine ;
- **FC3** : notre logiciel utilisera aussi la vérité terrain fournie afin d'associer les imagettes découpées à la bonne transcription ;
- **FC4** : il faudra également que celui-ci soit compatible avec le format des données qui lui seront fournies en entrée (format GEDI ou PiFF) ;
- **FC5** : enfin, notre logiciel devra regrouper les données afin de les stocker dans une base de données.

**Figure 7 : Diagramme pieuvre**



# Chapitre 5

## État de l'art

Notre projet est constitué de plusieurs briques élémentaires dont il convient de comprendre le fonctionnement et les alternatives qui existent. L'objectif de notre projet est de construire un logiciel générant des bases d'apprentissage pour des reconnaiseurs d'écriture manuscrite. Nous présenterons donc les différentes techniques permettant de réaliser cette tâche et en quoi consistent les techniques sur lesquelles sont basés les reconnaiseurs d'écriture. Le langage de programmation à utiliser dans le projet était libre : nous avons donc choisi le langage **Scala**, car il est un bon compromis entre la JVM, qui est notre zone de confort et qui permet d'utiliser toutes les bibliothèques Java dans le projet, et un paradigme fonctionnel, agréable à utiliser et qui permet d'introduire une touche supplémentaire d'originalité dans le projet. Les différentes technologies que nous avons étudiées font donc partie de l'écosystème de ce langage de programmation.

### 5.1 Techniques de reconnaissance d'écriture

Cette partie de l'État de l'art s'appuie sur les travaux de thèse de Luc MIOULET[5] sur les différents systèmes de reconnaissance d'écriture, et sur les travaux d'Adeline GRANET, Emmanuel MORIN, Harold MOUCHÈRE, Solen QUINIOU et Christian VIARD-GAUDIN sur la reconnaissance de documents historiques[2]. Le cadre de notre projet ne comprend pas le développement d'un reconnaisseur. Cette partie fera l'objet d'une collaboration entre Doptim et l'équipe IntuiDoc dans le cadre d'une thèse. Il est cependant intéressant de comprendre les mécanismes sur lesquels ceux-ci sont basés.

Afin de tester notre logiciel, nous utiliserons le système de reconnaissance Laia. Cet outil permet de créer des modèles de reconnaiseurs d'écriture manuscrite en *deep learning*. Il nous faudra donc utiliser leur modèle de reconnaisseur, l'entraîner sur une base d'apprentissage que nous aurons générée avec notre logiciel, et nous verrons si les résultats obtenus avec ce modèle sont bons ou non.

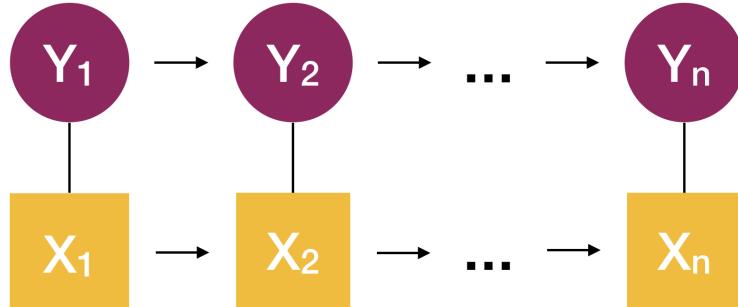
La plupart des techniques de reconnaissance de l'écriture sont basées sur des classificateurs dits dynamiques, c'est-à-dire qu'ils possèdent une mémoire interne ou possèdent une notion de contexte dans leur analyse. Ces classificateurs permettent un parcours et une division des données d'entrée et donc d'effectuer une classification pour chacune des divisions repérées.

#### 5.1.1 Modèles de Markov Cachés

Les Modèles de Markov Cachés (MMC) sont des modèles probabilistes basés sur une structure de graphe d'états. Ils modélisent une séquence basée sur des connaissances *a priori*, comme un langage pourrait être défini à l'aide de la forme des lettres. Les probabilités de passage d'un état à un autre dépendent d'une matrice de transition et un vecteur de conditions initiales permet de définir l'état de départ. Les MMC permettent de modéliser la probabilité de choisir un caractère en fonction de l'état actuel du système de reconnaissance. Pour cela, les différentes composantes de l'alphabet utilisé sont décomposées (première partie de boucle du d, barre verticale du 1, ...). Ces composantes correspondent aux états du graphe. On pourra donc modéliser la probabilité pour un système de choisir un 1 après avoir lu une barre verticale, etc. Ces probabilités ne seront pas égales à 1 car une barre verticale peut aussi correspondre à un t par exemple. Cependant, cette approche

impose une segmentation des composantes de l'alphabet que l'on souhaite traiter et impose de changer ces composantes ou d'étendre l'alphabet reconnu. Or, cette segmentation et l'étiquetage correspondant (e.g. "c'est une boucle d'un a") ne sont pas toujours évidents et peuvent être fastidieux à faire.

**Figure 8 : Schéma de transition et d'observation d'un MMC**

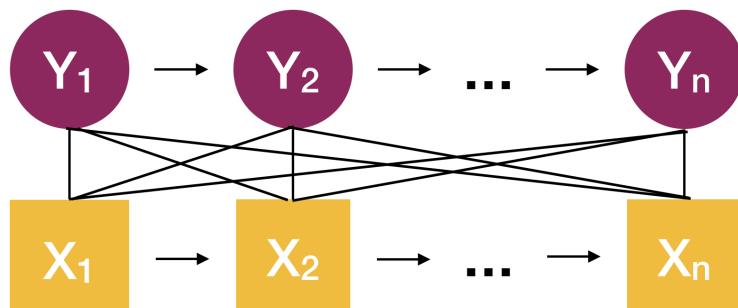


**Les Y représentent les états du système  
Les X sont les choix du reconnaiseur associé**

### 5.1.2 Champs Aléatoires Conditionnels

Les Champs Aléatoires Conditionnels sont des modèles basés également sur des probabilités et sur une structure similaire aux MMC. La principale différence entre ces deux méthodes est que pour les MMC, les probabilités de choix des caractères dépendent d'un état, tandis que pour les CAC, le choix d'un caractère peut être réalisé depuis n'importe quel état, avec une pondération par un potentiel. Cette approche permet alors de prendre en compte l'ensemble du contexte local de la séquence observée du fait de l'hypothèse de non-indépendance des états. Ce modèle est donc plus approprié pour l'analyse de séquences structurées. Cependant, ils n'effectuent pas d'étiquetage de séquence lors de leur apprentissage, et ne possèdent pas de structuration interne qui leur permettrait de saisir des structures de plus haut niveau telles que les titres ou la date dans un document.

**Figure 9 : Schéma de transition et d'observation d'un CAC**



**Les Y représentent les états du système  
Les X sont les choix du reconnaiseur associé**

### 5.1.3 Réseaux de Neurones Récursifs

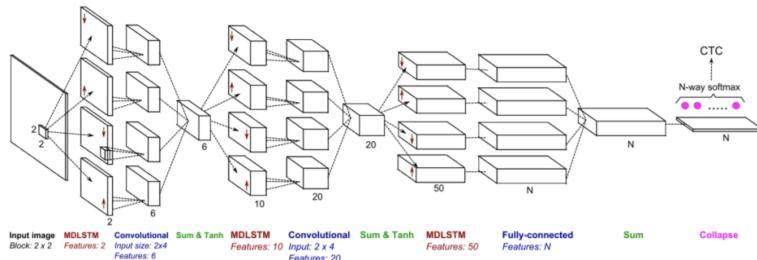
Les Réseaux de Neurones Récursifs sont basés sur la structure classique des réseaux de neurones, avec une modification : chaque neurone des couches cachées possède également en entrée les sorties des neurones correspondant à leur couche. Ceci permet d'avoir une mémoire interne des états de chaque couche. Via cette approche, il n'est pas nécessaire d'effectuer un étiquetage des données car le réseau apprend par lui-même.

Cette méthode apporte également une vue plus globale de l'ensemble d'entrée car on ne se limite pas à l'observation d'un état précédent. Le principal soucis de ces réseaux provient de l'apprentissage. Ils nécessitent un grand jeu de données et on remarque une "disparition" du gradient sur les couches les plus profondes, c'est-à-dire que les poids d'entrée ne sont que peu modifiés lors de l'apprentissage. Il faut alors recourir à un apprentissage couche par couche, ce qui est compliqué à cause de la récurrence de ce type de réseaux.

#### 5.1.4 Réseaux de Neurones Récursifs Multi-Dimensionnels

Les MDRNN sont conçus afin d'apporter une localité et une mémoire sur les deux dimensions de l'image traitée. Ils sont constitués de couches de réseaux de neurones de différentes natures organisés selon l'abstraction des données à effectuer. L'image est tout d'abord décomposée en blocs. Des réseaux récurrents la parcourront alors en la transformant dans chaque direction diagonale. Les sorties de ces réseaux sont alors transmises à un réseau convolutionnel qui permet d'augmenter le niveau d'abstraction des données de sortie. On fait ainsi circuler les résultats des différents réseaux vers d'autres afin d'extraire de plus en plus d'informations de haut niveau dans le document étudié.

Figure 10 : Schéma de structure d'un MDRNN



Cependant, cette structure ne permet pas une segmentation de l'image de départ. Il est donc nécessaire de créer cette segmentation afin d'établir une relation entre les données d'entrée et les sorties. Afin de réaliser cette segmentation, une adaptation des algorithmes d'apprentissage est nécessaire.

## 5.2 DéTECTEUR de lignes

Afin de pouvoir utiliser les techniques de reconnaissance manuscrite, il est d'abord nécessaire de faire une segmentation des documents numérisés que nous possédons en entrée. La méthode usuelle dans le domaine de la reconnaissance manuscrite est la segmentation par ligne des documents. Pour ce faire, plusieurs techniques sont documentées dans la littérature concernant l'OCR (*Optical Character Recognition*). Cette partie s'appuie notamment sur une revue des différentes techniques de segmentation parue dans l'[International Journal of Computer Science and Network Security](#) par neuf membres de l'université de Malaya[9]. La segmentation en lignes est une étape importante pour la reconnaissance manuscrite puisqu'un mauvais découpage va inévitablement poser des problèmes lors des étapes suivantes de la reconnaissance.

Il est alors important d'identifier les problèmes que pose la détection de lignes dans les documents manuscrits. En effet, certains aspects sont à prendre en compte :

- les lignes fluctuent au sein d'un même document ;
- les lignes d'un texte ne sont pas espacées régulièrement ;
- certaines lignes peuvent se chevaucher ;
- l'espace interligne lui aussi peut varier d'une ligne à une autre et d'un paragraphe à un autre ;
- les lignes d'un paragraphe peuvent avoir une courbure globale et une courbure locale propre à chaque ligne (et même locale à une partie d'une ligne).

---

Tous ces paramètres peuvent également différer entre deux paragraphes, deux textes, et deux écrivains. Nous allons maintenant présenter rapidement les principales techniques documentées dans la littérature de l'OCR, pour ensuite décrire plus précisément le fonctionnement des types de détecteurs de lignes que manipule l'équipe IntuiDoc. Nous tenons à préciser que les détecteurs de lignes que nous utiliserons nous seront fournis et que nous n'aurons pas à en implémenter un.

Il existe globalement deux types d'approches pour la détection des lignes dans un document manuscrit : les approches descendantes et les approches ascendantes. Les approches descendantes sont basées sur la technique de la projection. La projection horizontale est appliquée au document ou à une partie du document. Ensuite, une analyse sur les *maxima*, les *minima* et les composantes connexes entre deux *minima* consécutifs permettent de déterminer les lignes. Les approches ascendantes sont basées sur le bas niveau (au niveau pixel ou composantes connexes). La classification par K plus proches voisins (KNN), la transformée de Hough et la technique du lissage utilisent ce type d'approches.

Dans la classification par KNN, les alignements sont détectés en choisissant les composantes connexes, prolongées dans des directions spécifiques, puis les lignes sont extraites en groupant ces alignements selon ces critères : proximité, similitude, continuité de direction. Dans la transformée de Hough, les points sont les centres de gravité des composantes connexes. Un ensemble de points alignés dans l'image et ayant un pic dans la transformée de Hough représente une ligne. Dans la technique de lissage, les pixels noirs consécutifs sur la direction horizontale sont lissés. Les boîtes englobant les composantes connexes dans l'image lissée forment les lignes. Nous allons maintenant voir plus en détail les techniques qui nous intéressent dans le cadre de notre projet.

### 5.2.1 Détection de lignes par réseau de neurones à convolution

Pour la détection des lignes sur tout le document, un détecteur de lignes basé sur un réseau de neurones à convolution nous sera fourni par IntuiDoc. Cette partie s'appuie sur un article[1] de Charles CROUSPEYRE à propos du fonctionnement des réseaux convolutifs, et sur les travaux de Sofia ARES OLIVEIRA, Benoît SEGUIN, et Frédéric KAPLAN[7] sur l'approche *deep learning* de la segmentation de documents.

Les réseaux de neurones à convolution comparent les images fragment par fragment et recherchent des fragments caractéristiques. Par exemple, un réseau de neurones à convolution cherchant à déterminer si une image est un X ou pas va chercher les caractéristiques définissant les X, les diagonales et leur entre-croisement. Lorsqu'on lui présente une image, le système ne sait alors pas si les caractéristiques sont présentes ni leur localisation. Il va alors calculer dans toute l'image si la caractéristique est présente. Pour compléter une convolution, on répète ce processus alignant les caractéristiques à chaque sous-partie de l'image. Le résultat de chaque convolution est une image stockant le résultat de la comparaison, avec un 1 pour une similitude et un -1 pour une différence par exemple. Ensuite, à partir du résultat de chaque convolution, on crée un tableau situant où les caractéristiques se trouvent dans l'image. On obtient alors des images correspondant à l'image de base découpée en filtres. On répète le processus de convolution pour chaque caractéristique, et on obtient alors la couche de convolution.

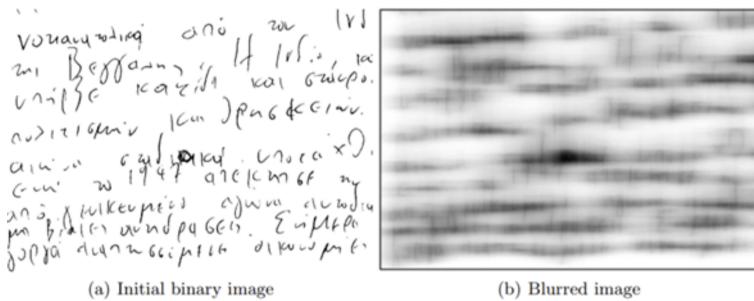
Cependant, le nombre d'opérations augmente linéairement avec le nombre de pixels de l'image et le nombre de pixels de chaque caractéristique. On peut alors effectuer du *pooling*. On va réduire le nombre de pixels de l'image, en passant une fenêtre de quelques pixels sur l'image, et dans laquelle on ne garde que la valeur qui nous importe. Ainsi, le réseau ne situe pas précisément les caractéristiques dans l'image, il connaît seulement leur présence ou non. La couche de *pooling* sert simplement à diminuer la charge de calcul. En effet, on garde le même nombre d'images avec les mêmes informations, mais le nombre de pixels est moindre.

### 5.2.2 Détection de ligne par floutage

Pour détecter les lignes sur un paragraphe, un détecteur de lignes par floutage sera utilisé. Cette partie s'appuie sur les travaux d'Aurélie LEMAITRE, Jean CAMILLERAPP, et Bertrand COÜASNON[4] sur la segmentation de textes manuscrits par l'utilisation d'images floutées. Cette technique utilise la combinaison de deux niveaux d'analyse de l'image. Tout d'abord, une première analyse bas niveau de l'image floutée est réalisée,

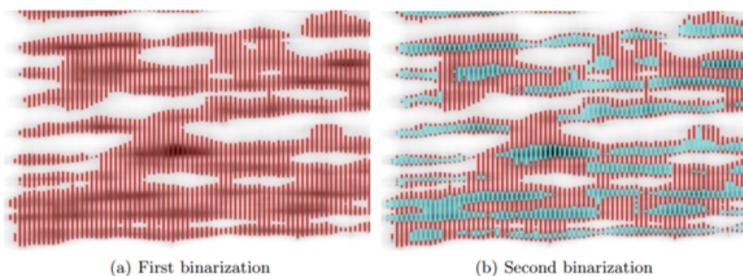
ce qui est efficace pour les images à forte densité textuelle. Ensuite, une analyse plus haut niveau est effectuée, ce qui est efficace sur des images à faible densité textuelle. La combinaison des deux niveaux d'analyse permet de gérer les problèmes liés à la détection de lignes dans les documents manuscrits évoqués au début de ce chapitre. L'analyse bas niveau de l'image floutée est composée de deux étapes que sont l'extraction de régions noires dans l'image afin de construire des parties de l'axe de la ligne, ainsi que la construction de l'axe de la ligne.

**Figure 11 : Floutage d'une image**



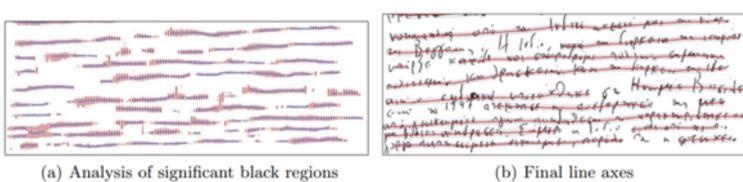
Afin d'extraire les régions noires correspondant à des parties d'une ligne, il est nécessaire d'effectuer préalablement un traitement sur l'image. Tout d'abord, l'image est floutée, puis une double binarisation lui est appliquée. Le résultat est une image dans laquelle sont en évidence les « régions noires » qui vont être utilisées pour construire l'axe des lignes.

**Figure 12 : Binarisation d'une image**



Ces traitements d'image et la localisation de ces « zones noires » vont permettre ensuite de construire les axes des lignes. Une analyse de la position et de l'épaisseur de ces zones va permettre de les relier entre elles. On pourra alors calculer une épaisseur moyenne et une pente moyenne pour le document, données qui vont ensuite servir pour étendre les axes.

**Figure 13 : Extension des axes correspondant aux lignes de texte**



---

En effet, la dernière opération de cette étape consiste à trier les différents axes créés. Des règles prenant en compte l'épaisseur, la position et la pente des axes vont déterminer quels axes fusionner et quels axes supprimer. On peut obtenir des axes incomplets en résultat de cette première analyse bas niveau. Cependant, aucune analyse supplémentaire ne peut être réalisée à ce niveau.

Le deuxième niveau d'analyse prend en compte le modèle du document étudié. À partir des différents axes de lignes créés dans l'étape précédente, le but est de créer les lignes du texte. Ainsi, en fonction du document étudié, les stratégies de construction des lignes seront différentes. Afin de spécifier un document, une description grammaticale basée sur deux types d'entités (les éléments connectés et les axes créés précédemment) est effectuée. La description grammaticale du modèle du document va permettre de localiser les lignes du texte et d'assigner des pixels à des lignes. Grâce à la description grammaticale et à des règles judicieusement choisies, on peut alors construire les lignes du document. Cependant, il peut-être nécessaire (selon le domaine d'application) d'associer chaque pixel à sa ligne. Une dernière analyse basée sur la description grammaticale du document couplée à certaines informations sur le document est effectuée pour réaliser l'association.

Davantage d'informations sur ce sujet compliqué peuvent être obtenues en étudiant l'article[3] donné en bibliographie.

## 5.3 Format de description d'image

### 5.3.1 GEDI

Le logiciel GEDI est un outil qui permet d'annoter des documents scannés. Il est ainsi très utile pour établir la vérité terrain. Il met en scène deux types de documents : des images qui correspondent aux documents scannés ainsi que des documents dans un format dérivé du XML auquel il donne son nom (le format GEDI), qui permettent de stocker toutes les informations relatives aux documents scannés. On peut alors avoir, pour chaque document scanné, des informations relatives à la position des paragraphes, à la langue dans laquelle le texte est écrit, ou encore à la forme du texte (manuscrit ou imprimé). La vérité terrain que nous aurons au sein de notre projet aura été établie avec GEDI.

### 5.3.2 PiFF

Le Pivot File Format (PiFF) est un format de description d'image basé sur JSON et créé par différents chercheurs français : Harold MOUCHÈRE, Christopher KERMORVANT, Andres ROJAS, Mickal COUSTATY, Joseph CHAZALON et Bertrand COÜASNON. C'est un format intermédiaire préalable au traitement d'une image dans le cadre d'analyses de documents. Ce format est très utile pour l'analyse de documents puisqu'il permet le partage de jeux de données, le traitement de résultats ainsi que l'utilisation d'outils déjà existants sans avoir à faire de conversion entre les différents formats qui pourraient exister. De ce fait, les différentes étapes de l'analyse de documents peuvent être effectuées par différentes équipes sans qu'il n'y ait de conflit au niveau du format des données, ce qui permet une collaboration plus facile. Les données présentes en entrée dans le cas de notre projet pourront se trouver sous différents formats, soit au format PiFF et dans ce cas il n'y aura rien à changer, soit dans un autre format quelconque et il faudra alors les convertir. Les données de la base Maurdor étant au format GEDI, il nous faudra notamment un convertisseur GEDI/PiFF. De ce fait, les données après traitement par le logiciel seront toutes dans le format PiFF, ce qui permettra de les utiliser facilement.

## 5.4 Base de données

La question de la base de données est une question qui a été soulevée puisqu'elle constitue le pivot de notre logiciel. En effet, elle centralise les couples imagette-retranscription produits lors du traitement des données et est utilisée par tous les autres services. Tout d'abord se pose la question du stockage des images dans une base de données. Il existe plusieurs façons de le faire. Il est possible de stocker l'image directement dans une base de données SQL en binaire. Cependant, cette étape demande de transformer l'image pour l'insérer et de nouveau

---

pour la charger, ce qui peut poser des problèmes de latence s'il s'agit de grosses images ou s'il y a beaucoup d'images à charger. Un autre moyen qui existe est de stocker non pas l'image mais une référence vers l'image, soit son *path*. Cette méthode, répandue en entreprise, a l'avantage de ne pas surcharger la base de données et est souvent recommandée. Même si alors le poids de l'image ne pose plus problème, on n'accède plus à l'objet image en lui-même, et cela impose de les stocker dans un serveur de fichiers hors de la base. Il existe également d'autres solutions n'utilisant pas les bases SQL. En effet, les bases de type NoSQL permettent de stocker les images telles quelles du fait de leur orientation document (on constitue des collections de documents que l'on lie). Ainsi, il serait possible d'intégrer les images dans la base sans devoir les traduire dans un autre format.

## 5.5 Interface Homme-Machine

Les technologies existantes pour réaliser une IHM sont abondantes. Deux lignes de conduite s'offrent à nous. Nous pouvons soit concevoir une application "lourde" comprenant une interface logicielle construite avec un outil du type de [JavaFX](#), soit créer une interface web. Nous devons gérer une base de données et nous souhaitons que notre logiciel soit portable et puisse fonctionner sur un serveur. Nous avons donc préféré nous orienter vers une implémentation web de l'interface. Cependant, il existe de multiples moyens d'utiliser une base de données avec une application web. Nous pouvons par exemple utiliser des requêtes [REST](#), gérer la base de manière plus proche avec des scripts de type NodeJS ou PHP et construire l'interface en HTML/CSS et JS. Il existe également des bibliothèques basées sur le modèle MVC, d'autres non. Nous avons isolé quatre frameworks web basés sur Scala : [Scala.js](#), [Play](#), [Chaos](#) et [Lift](#). Scala.js est rapide, fiable, facile à utiliser et utilise les bibliothèques JS existantes et une multitude d'autres bibliothèques standard. Play est rapide, utilise [Maven](#) et la JVM, possède une communauté très active et propose un rendu en temps réel. Chaos, très utilisé pour les applications REST, est léger, facile à prendre en main et utilise la JVM, Scala et [Jersey](#). Cependant, il n'est performant que pour les applications REST. Enfin, Lift est principalement orienté vers les applications avec un fort besoin de sécurité. Il est rapide, facile à maintenir et propose des modules pré-construits ainsi qu'une large documentation dont la qualité est néanmoins très variable. Ainsi, il nous est nécessaire de définir les rôles et les objectifs de l'interface ainsi que les interactions qu'elle aura avec le reste du programme afin de pouvoir déterminer l'approche et les technologies nécessaires.

## 5.6 Traitement d'images

Pour le traitement d'images, plusieurs bibliothèques logicielles sont à notre disposition. On peut citer [Pillow](#), [ImageJ](#), [ImageMagick](#) ou [OpenCV](#). Cette dernière nous a été conseillée par notre encadrant de projet, et elle est reconnue comme un standard parmi les technologies de traitement d'images utilisées par la communauté, car elle est disponible sur plusieurs plateformes et interfaçable avec plusieurs langages (C++, Java, Python, ...). Elle convient donc parfaitement aux besoins que nous avons (compatibilité avec la JVM, bibliothèque complète et *cross-platform*). D'après le cahier des charges, nous n'aurons pas à utiliser ses fonctionnalités les plus avancées (à base d'intelligence artificielle) au cours de ce projet, mais cela permet une fois de plus que le projet soit évolutif, c'est-à-dire que les chercheurs qui maintiendront le projet pourront y ajouter des éléments de traitement d'images plus avancés. Ce choix est donc validé dès maintenant.

# Chapitre 6

# Organisation du projet

## 6.1 Méthodes de travail

Cette année, nous avons décidé d'utiliser une méthode de travail suivant un cycle en V. Cette méthode nous semble la plus adaptée pour notre cas. En effet, les limites du projet sont clairement définies, de même que le cahier des charges et les spécifications générales. Ainsi, durant la première phase du projet, au premier semestre, nous allons mettre en place des réunions régulières pour bien évaluer le travail à réaliser. Durant la seconde phase, au second semestre, nous effectuerons des réunions pour voir l'avancée de la programmation.

Comme le projet peut facilement être découpé en 4 parties (Préparation des données, Base de données, IHM et Gestion du reconnaiseur), nous avons décidé de diviser notre groupe en 4 pour pouvoir faire avancer le projet en parallèle. Nous estimons qu'il est tout de même important de garder une forte communication et cohésion entre les différents groupes, donc nous avons mis en place différents systèmes de communication. Pour gérer l'organisation, nous utiliserons Trello et Microsoft Project. Pour le partage de fichiers, notamment pour la rédaction des rapports, nous utiliserons Google Drive et Git. Pour la gestion du code, nous utiliserons aussi Git. Enfin, pour communiquer, nous utiliserons une conversation Messenger (qui servira à transmettre des informations de manière rapide), un groupe Facebook (qui servira à publier la répartition des tâches et les objectifs de chaque semaine de manière fixe et accessible) et probablement des salons Discord qui permettront à chaque groupe de communiquer sans perturber les autres groupes.

Pour l'instant, nous avons décidé d'établir des réunions hebdomadaires (voire deux réunions par semaine en cas de retard ou de livrables à rendre). Cette planification sera probablement modifiée au second semestre lors du début du développement, car le groupe sera réduit de moitié à cause des départs en mobilité. En effet, à partir du mois de janvier et du début du second semestre, certaines personnes partiront faire une mobilité à l'étranger et ne travailleront donc plus sur le projet. Notre groupe sera amputé de trois personnes : Kevin DESPOULAINS, Gaël GENDRON et Corentin GUILLOUX.

L'arrivée ou le départ de membres n'est pas rare dans le déroulement d'un projet et nous avons réfléchi à une organisation qui permet de faire une transition souple entre le premier semestre où nous sommes huit et le second où nous serons cinq. Les membres partant à l'étranger ne travailleront en effet pas seuls sur une partie mais leur travail sera réparti sur plusieurs aspects du projet où ils travailleront avec des personnes restant au second semestre. Cette répartition est faite pour qu'au moment de leur départ, il y ait au moins une personne du groupe restant sachant exactement ce qui a été fait par les étudiants qui sont partis.

## 6.2 Répartition des tâches

Afin de répondre aux besoins du projet, nous avons mis en place 3 rôles. Ces 3 rôles ne sont pas fixes, ils peuvent être modifiés en cas de besoin. Le premier rôle est le chef de projet. Ce rôle n'est donc pas assigné de manière fixe, il change régulièrement. Nous avons établi un calendrier précis en figure 14 qui donne l'indication de qui est chef de projet à chaque instant.

**Figure 14 : Planning des rotations du rôle de chef de projet**

Chef de projet	Début	Fin
Kevin DESPOULAINS	17/09/2018	07/10/2018
Gaël GENDRON	08/10/2018	28/10/2018
Corentin GUILLOUX	05/11/2018	30/11/2018
Valentin FOUCHER	01/12/2018	13/01/2019
Enzo CRANCE	14/01/2019	04/02/2019
Charlotte RICHARD	05/02/2019	10/03/2019
Laure DU MESNILDOT	11/03/2019	05/04/2019
Timothée NEITTHOFFER	23/04/2019	10/05/2019

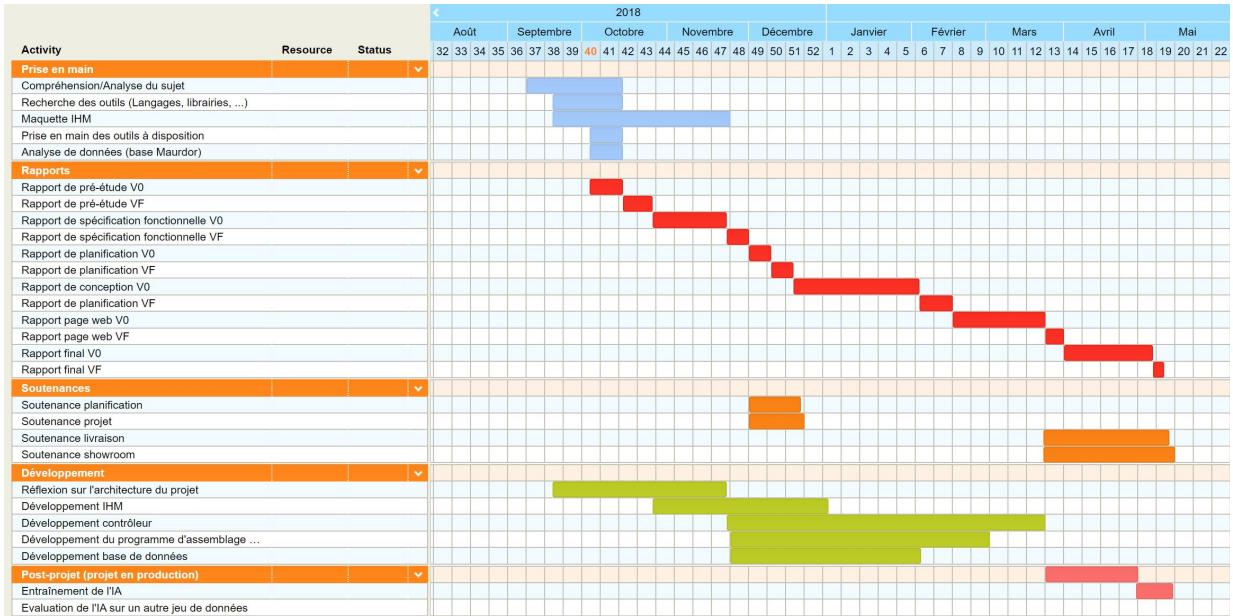
Septembre	Octobre	Novembre	Décembre	Janvier	Février	Mars	Avril	Mai
					Rapport conception Version 0			Rapport final Version 0
			Rapport planification Version 0					Rapport final Version Finale Soutenances
	Rapport pré-étude Version 0				Rapport conception Version Finale			
			Rapport planification VF					
		Rapport spécifications Version 0	Soutenances planification et projet			Page web Version 0		
	Rapport pré-étude Version Finale							
		Rapport spécifications Version Finale				Page web Version Finale		

Certaines périodes peuvent paraître plus longues pour certains que pour d'autres à cause des vacances scolaires, mais chacun sera chef de projet durant 3 à 4 semaines. Le rôle du chef de projet est avant tout de s'assurer du bon déroulement du projet. Il doit organiser les réunions, répartir équitablement les tâches, surveiller les délais et diriger les réunions. Il doit également faire le lien avec les différents intervenants et contrôler ce qui est écrit dans les rapports. Le deuxième rôle est celui de responsable temps qui devra s'assurer que tout le groupe note bien le temps passé sur le projet en envoyant des rappels réguliers. Il devra également gérer le temps de parole lors des réunions. Pour le moment, c'est Timothée NEITTHOFFER qui a été désigné responsable temps. La personne jouant ce rôle peut être modifiée sans problème au cours de l'année. Enfin, le dernier rôle est responsable qualité. Son but est de contrôler la qualité du code réalisé. Au premier semestre, c'est lui qui valide ou non les technologies que les autres membres du groupe peuvent lui suggérer en les étudiant. Au second semestre, il devra contrôler le code. Pour le moment, c'est Enzo CRANCE qui a été désigné responsable qualité. Les autres membres du groupe travaillent davantage sur les tâches que le chef de projet leur a confié comme la rédaction des rapports, la recherche d'information, le développement, etc.

## 6.3 Estimation de la planification des tâches

Pour avoir une première estimation du planning, nous avons construit un diagramme de Gantt en figure 15.

**Figure 15 : Estimation de la planification des tâches**



**Prise en main du projet depuis sa compréhension et son analyse à la rédaction du cahier des charges. Cette partie se fait essentiellement en début de projet.**

**Rédaction des différents livrables.** Nous avons estimé que le début de rédaction d'un livrable doit se faire dès le rendu du livrable précédent.

**Soutenances.** Les dates de début de préparation sont pour l'instant difficiles à définir, la taille des blocs est donc provisoire.

**Développement.** Comme on peut le voir, plusieurs parties du développement se font en parallèle. Les durées de chaque partie sont prévisionnelles mais sont susceptibles d'être grandement modifiées.

**Exécution du projet en mode production.** Ce seront les tests du reconnaissleur.

# Chapitre 7

## Conclusion

Dans le cadre de notre 4<sup>ème</sup> année en informatique, nous avons pour mission de mener à bien un projet de manière plus précise et plus structurée qu'en 3<sup>ème</sup> année. Notre projet, en partenariat avec les archives départementales d'Ille-et-Vilaine, l'équipe IntuiDoc et Doptim, consiste à créer un logiciel qui aidera les chercheurs et les ingénieurs à faire progresser la reconnaissance d'écriture manuscrite, afin de rendre des documents anciens plus accessibles et plus compréhensibles.

Notre groupe se compose de 8 étudiants : Enzo CRANCE, Kevin DESPOULAINS, Timothée NEITTHOF-FER, Laure DU MESNILDOT, Charlotte RICHARD, Valentin FOUCHER, Gaël GENDRON et Corentin GUILLOUX. 3 d'entre nous partiront en mobilité internationale au second semestre : Kevin DESPOULAINS, Gaël GENDRON et Corentin GUILLOUX. Il nous faudra ainsi prendre en compte dans la réalisation de notre projet ce changement d'effectif. L'étude du projet, la répartition des tâches et la planification ont donc été faites bien en amont des phases de développement pour permettre à ceux qui seront encore présents au second semestre de ne pas prendre de retard.

Nous avons dans un premier temps étudié ce qui nous était demandé de faire avant de décider des technologies que nous allions utiliser. Nous avons bien sûr commencé à étudier les différents outils (langages, API, etc.) que nous pourrions utiliser et qui feront l'objet du prochain rapport.

Nous avons ensuite rédigé notre cahier des charges en reprenant les différentes fonctionnalités que nous souhaitons développer, en accord avec Bertrand COÜASNON, tout en prenant en compte les éléments extérieurs avec lesquels notre programme doit communiquer.

Nous avons également distribué les rôles qui nous semblent importants pour veiller au bon déroulement du projet sans mettre trop de pression sur les personnes les endossant. Un planning prévisionnel a également été rédigé.

# Bibliographie

- [1] Charles CROUSPEYRE. Comment les réseaux de neurones à convolution fonctionnent. <https://medium.com/@CharlesCrouspeyre/comment-les-reseaux-de-neurones-%C3%A0-convolution-fonctionnent-b288519dbcf8>, 2017. Traduction d'un article de Brandon Rohrer, *data scientist* chez Facebook.
- [2] Adeline GRANET, Emmanuel MORIN, Harold MOUCHÈRE, Solen QUINIOU, and Christian VIARD-GAUDIN. Étude préliminaire de reconnaissance d'écriture sur des documents historiques. [http://www.asso-aria.org/coria/2017/RJCRI\\_11.pdf](http://www.asso-aria.org/coria/2017/RJCRI_11.pdf), 2017.
- [3] Honglak LEE, Roger GROSSE, Rajesh RANGANATH, and Andrew Y. NG. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. <http://web.eecs.umich.edu/~honglak/icml09-ConvolutionalDeepBeliefNetworks.pdf>, 2009. In Proceedings of the 26th International Conference on Machine Learning. Montreal, Canada, 2009.
- [4] Aurélie LEMAITRE, Jean CAMILLERAPP, and Bertrand COÜASNON. Handwritten text segmentation using blurred image. <https://hal.inria.fr/hal-01087210/document>, 2014. DRR - Document Recognition and Retrieval XXI, Jan 2014, San Francisco, United States.
- [5] Luc MIOULET. Reconnaissance de l'écriture manuscrite avec des réseaux récurrents. traitement du texte et du document. <https://hal.archives-ouvertes.fr/tel-01301728/document>, 2015.
- [6] Harold MOUCHÈRE, Christopher KERMORVANT, Andres ROJAS, Mickal COUSTATY, Joseph CHAZALON, and Bertrand COÜASNON. Piff : a pivot file format, 2017. In Proceedings of the 1st International Workshop on Open Services and Tools for Document Analysis. Kyoto, Japan, IEEE.
- [7] Sofia ARES OLIVEIRA, Benoît SEGUIN, and Frédéric KAPLAN. dhsegment : A generic deep-learning approach for document segmentation. <https://arxiv.org/abs/1804.10371>, 2018.
- [8] Joan PUIGCERVER, Daniel MARTIN-ALBO, and Mauricio VILLEGAS. Laia : A deep learning toolkit for htr. <https://github.com/jpuigcerver/Laia>, 2016. GitHub repository.
- [9] Zaidi RAZAK, Khansa ZULKIFLEE, Mohd Yamani Idna IDRIS, Emran Mohd TAMIL, Mohd Noorzaily Mohamed NOOR, Rosli SALLEH, Mohd YAAKOB, Zulkifli Mohd YUSOF, and Mashkuri YAACOB. Off-line handwriting text line segmentation : A review. [http://paper.ijcsns.org/07\\_book/html/200807/200807003.html](http://paper.ijcsns.org/07_book/html/200807/200807003.html), 2008. IJCSNS - International Journal of Computer Science and Network Security, VOL.8 No.7, July 2008.

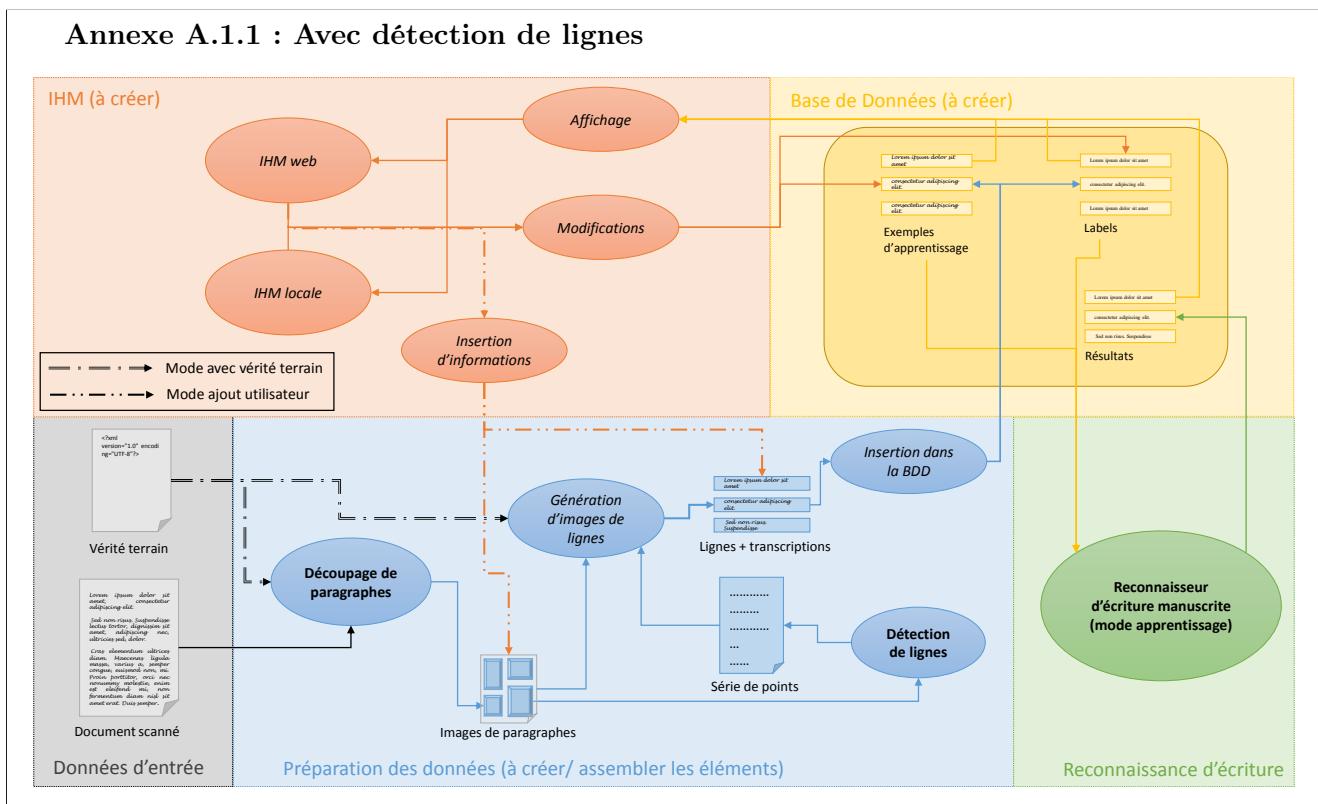
# Chapitre 8

## Annexes

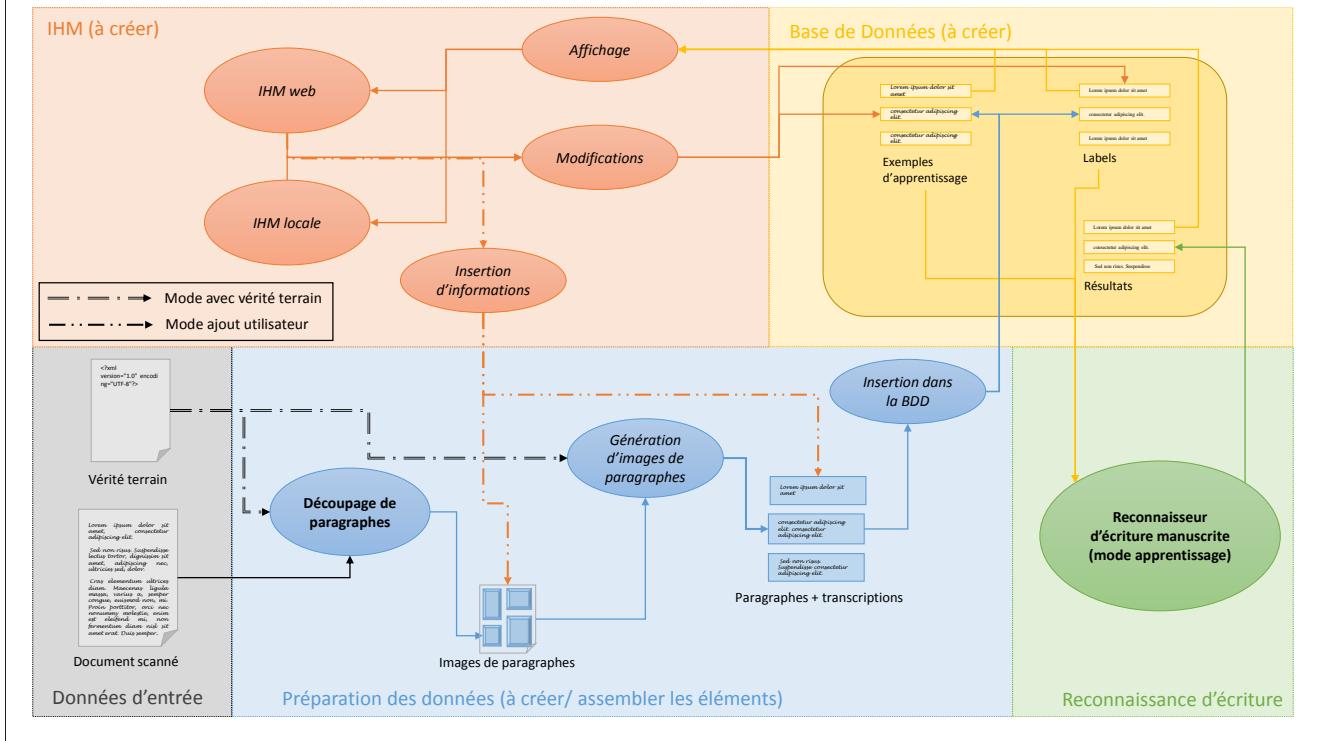
### 8.1 Annexe A : Schémas résumant le projet dans chacun de ses modes de fonctionnement

#### 8.1.1 Mode apprentissage

Dans le mode apprentissage, le logiciel doit permettre à l'utilisateur de rentrer dans la base de données des documents manuscrits avec une vérité terrain ou non. Dans le second cas, l'utilisateur rentre à la main une transcription à partir de l'IHM. La base ainsi créée est utilisée pour permettre un apprentissage du reconnaiseur.



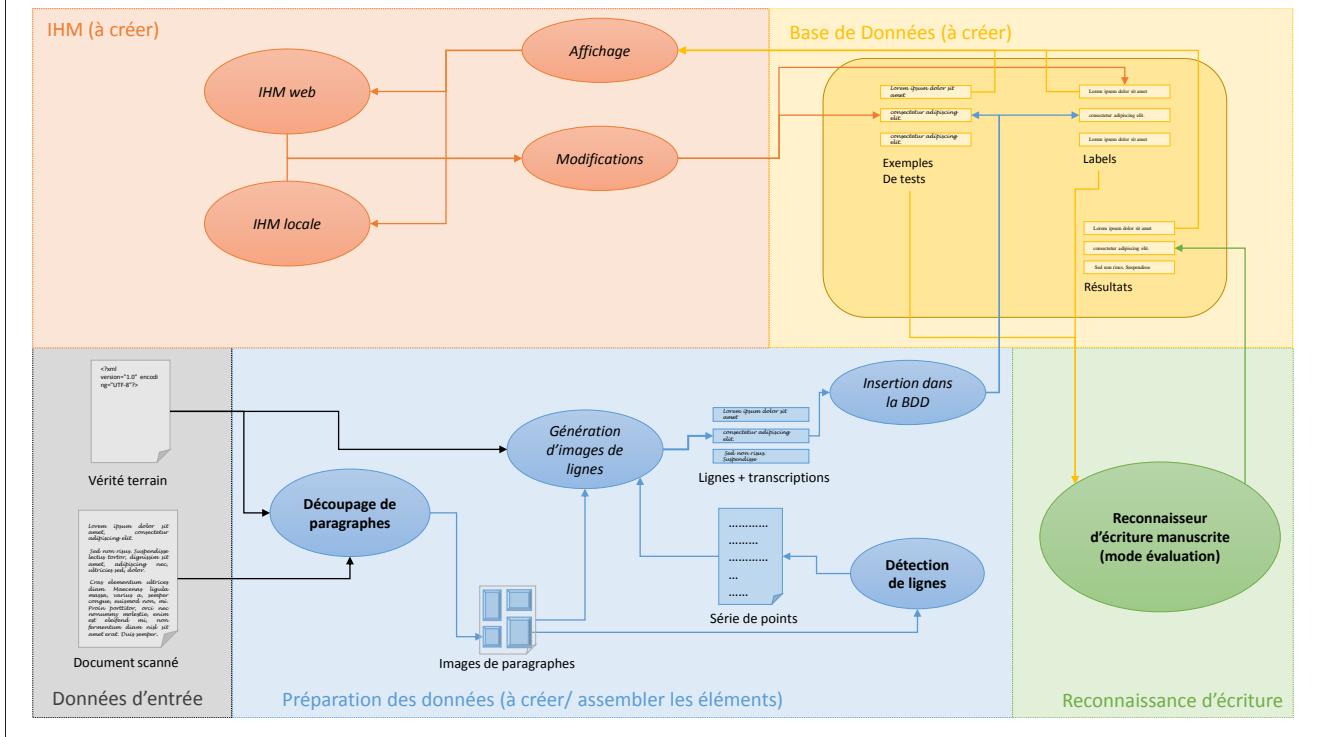
## Annexe A.1.2 : Sans détection de lignes



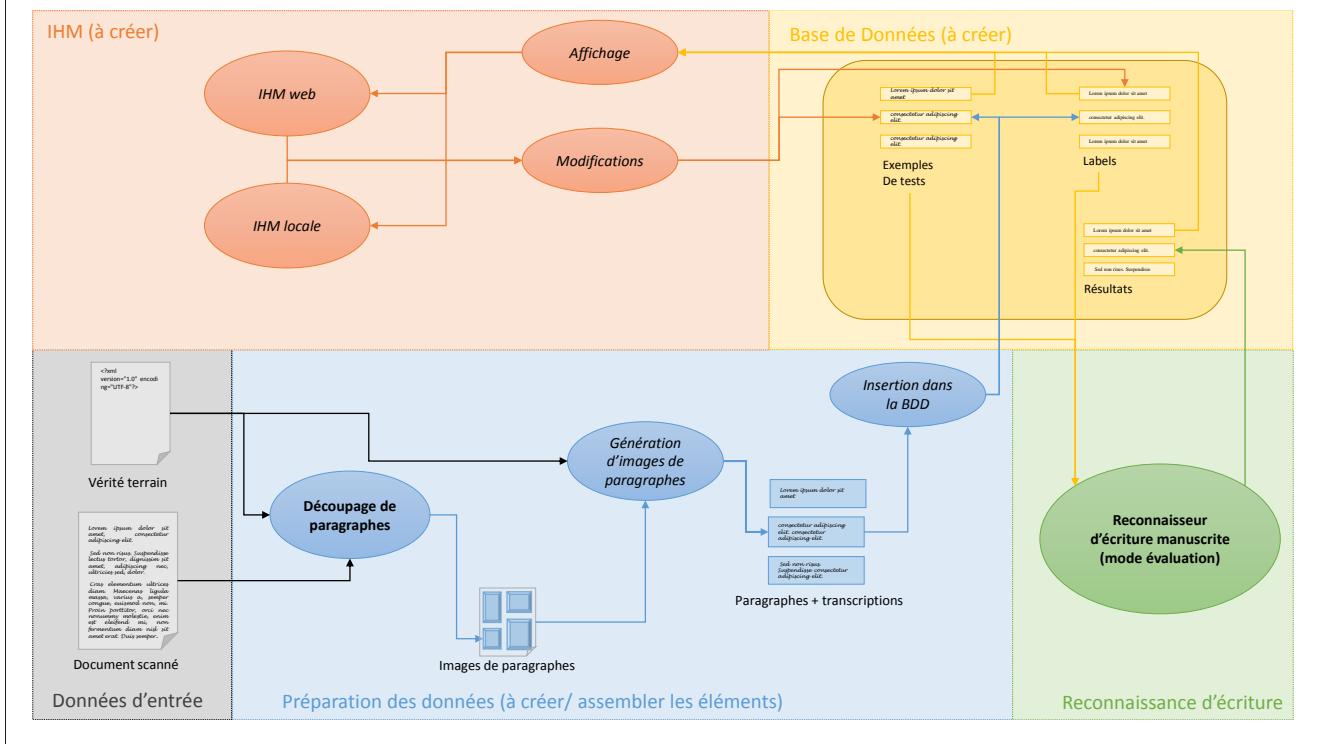
### 8.1.2 Mode évaluation

Dans le mode évaluation, le logiciel doit permettre à l'utilisateur de rentrer des documents dans la base de données avec une vérité terrain. La base créée est utilisée pour évaluer le reconnaiseur et vérifier son efficacité.

## Annexe A.2.1 : Avec détection de lignes



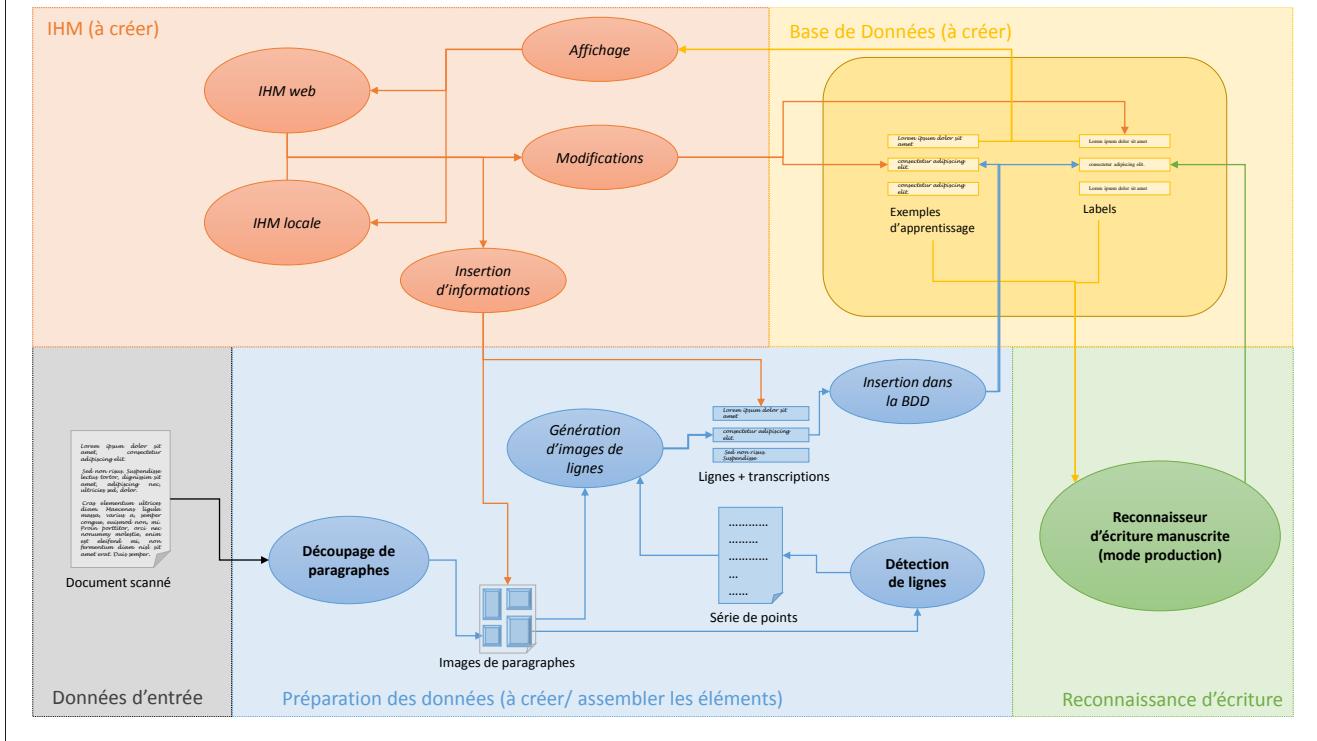
## Annexe A.2.2 : Sans détection de lignes



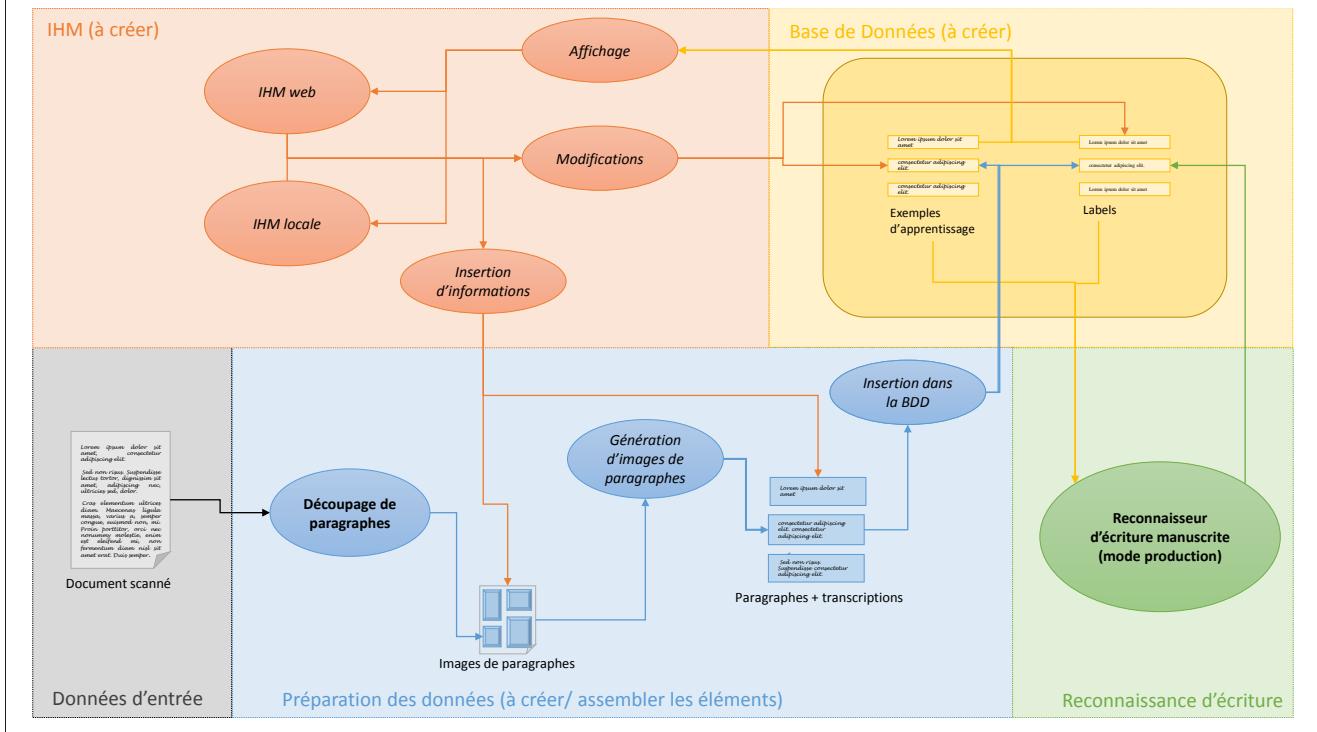
### 8.1.3 Mode production

Dans le mode production, le logiciel doit permettre à l'utilisateur de rentrer des documents sans vérité terrain ni transcription. Le reconnaiseur fournit alors une retranscription du document. Ce mode peut être utilisé pour plusieurs cas d'utilisation : la transcription fournie par le reconnaiseur est complétée par l'utilisateur et permet de transcrire rapidement un document numériquement. Il peut aussi être utilisé pour effectuer des démonstrations.

### Annexe A.3.1 : Avec détection de lignes



### Annexe A.3.2 : Sans détection de lignes





**INSA Rennes**  
20 Avenue des Buttes de Coësmes  
CS 70839  
35708 Rennes Cedex 7  
Tél. +33 [0] 2 23 23 82 00  
Fax +33 [0] 2 23 23 83 96

[www.insa-rennes.fr](http://www.insa-rennes.fr)

**INSA**

UNIVERSITE  
BRETAGNE  
LOIRE

**Cti**  
Commission  
des Titres d'Ingénieur

