Le pipeline du projet *Gallic(orpor)a* se déroule dans cinq étapes. En premier temps, il récupère les fac-similés numériques des documents sources sur Gallica. En deuxième temps, il applique des modèles HTR aux fac-similés téléchargés afin de produire une prédiction du texte et une transcription de la mise en page. Ensuite, il crée un fichier TEI préliminaire qui réunit les données produites par les modèles HTR et les métadonnées récupérées de plusieurs sources en ligne. Le quatrième étape enrichit le fichier TEI avec une analyse linguistique du texte de la transcription. En fin, le pipeline export les données du fichier TEI en divers formats, y compris les données conformant aux schèmes RDF, DTS, et IIIF.

1 La récupération des fac-similés numériques

L'accès aux pages numérisées d'un document source est une condition essentielle à toute étape du pipeline de *Gallic(orpor)a*. Afin de transcrire le document et produire la ressource lexicographique du format TEI, il faut d'abord dire au logiciel comment il peut accéder au fac-similé numérique. De plus, les images doivent être d'une qualité aussi bonne, sans être pourries par trop de bruit, que le logiciel HTR peut bien reconnaître le texte et les segments dedans. Et en fin, bien qu'il ne soit pas essentiel à l'étape de la transcription de l'image, l'image devrait s'associer aux métadonnées qui porte sur le document pour que le pipeline puisse enrichir la ressource lexicographique avec les informations portant sur le document transcrit.

1.1 Archival Resource Key (ARK)

Deux solutions existent pour répondre aux questions d'accès et de métadonnées. En premier temps, il existe dans le monde archivistique une espèce de clef numérique qui se connaît par l'acronyme ARK, qui veut dire en anglais *Archival Resource Key*. Cette clef est un identifiant unique qui indique une ressource, soit numérique soit physique, dont la conservation une institution, telle que la Bibliothèque nationale de France, se charge. Le schème ARK est actuellement maintenu par la communauté ARK Alliance. [1] L'identifiant unique facilite la récupération d'une ressource en particulière, sans la confondre avec d'autres exemplaires du même document. La précision qu'accorde l'ARK améliore l'exactitude d'un processus de traitement automatique.

1.1.1 La mise en place d'un système de fichiers

Le pipeline se sert de l'identifiant ARK afin de gérer le lien entre les images numériques et le document source. Une partie du système de fichiers du pipeline de *Gallic(orpor)a* est visualisée dans la Figure 1, qui montre l'exemple de deux documents sources dans lequel chacun a trois pages numérisées en format JPEG.

1. The ARK Alliance. *Community*. ARK Alliance. 6 nov. 2020. URL : `https://arks.org/community/` (visité le 23/08/2022).

Le chemin d'accès à chaque image se compose donc de l'identifiant ARK. Ainsi son association au document source est conservée et accessible au logiciel.

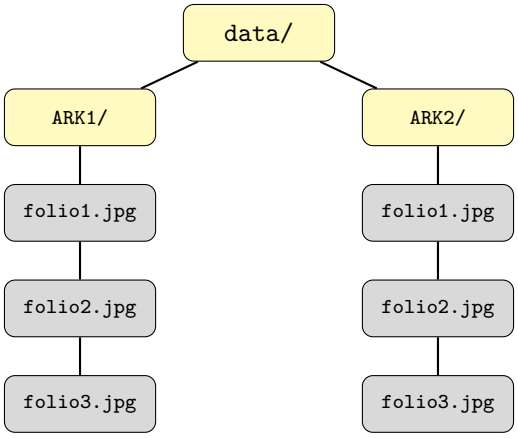


FIGURE 1 – Système de fichiers

1.2 International Image Interoperability Framework (IIIF)

En plus d'associer les images à leur document source, l'identifiant ARK sert aussi à récupérer les images depuis l'internet. Ce genre de requête se fait par un outil généralisé qui s'appelle une API (*Application Programming Interface*). Une telle interface facilite la communication entre deux ordinateurs pour qu'ils puissent échanger d'information. Ainsi, un ordinateur peut demander aux serveurs de la Bibliothèque nationale de France les images d'un document source qu'ils conservent.

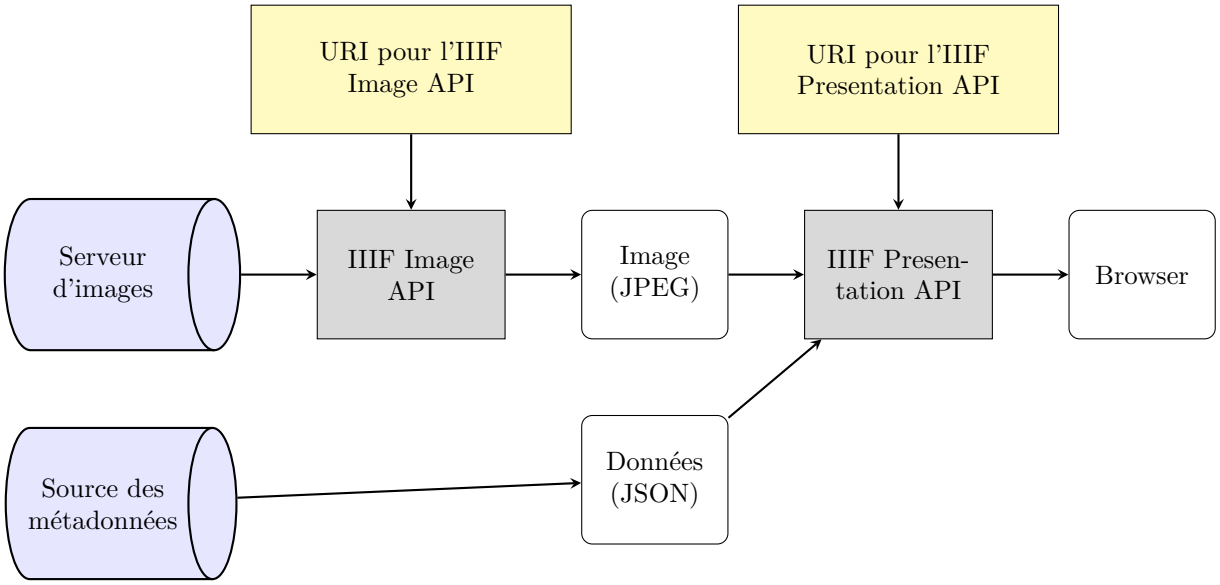
Comme des autres institutions participantes, la BnF met en pratique une API spécialisée à l'échange des données visuelles qui vient d'une initiative internationale qui s'appelle *l'International Image Interoperability Framework*. Le IIIF s'engage à standardiser la gestion des ressources visuelles mises en ligne avec le but d'améliorer l'exploitation et la partage des ressources et de leurs métadonnées. Suite à une requête standardisée HTTP(S), l'IIIF Image API renvoie l'image. L'URI se compose de quatre éléments généralisés, suivis par cinq éléments qui peut préciser (1) la région, (2) la taille, (3) la rotation, (4) la qualité, et (5) le format de l'image demandée. Puisque la BnF s'engage à l'initiative de l'IIIF, toute image sur sa base de données Gallica peut être requêtée par l'URL que l'IIIF Image API attend.

scheme	`https://`
server	`gallica.bnf.fr`
prefix	`/iiif`
identifier	`/ark:12148/bpt6k1057726c`
folio	`/f1`
region	`/full`
size	`/full`
rotation	`/0`
quality	`/native`
format	`.jpg`

(a) URI pour l'IIIF Image API

scheme	`https://`
server	`gallica.bnf.fr`
prefix	`/iiif`
identifier	`/ark:12148/bpt6k1057726c`
manifest	`/manifest`
format	`.json`

(b) URI pour l'IIIF Presentation API



(c) IIIF Image API et IIIF Presentation API

FIGURE 2 – IIIF APIs

La Figure 2 montre comment construire les URI qui se servent de la requête à une API IIIF. Prenons le document source de l'identifiant ARK bpt6k1057726c. Pour récupérer ses pages numérisées depuis la base de données Gallica, on envoie une requête à l'IIIF Image API ; l'exemple pour récupérer la totalité de la première page du document en format JPEG, sans rotation ni d'autre modification, se voit dans la Figure 1a. Comme la Figure 1c visualise, une telle requête HTTPS envoyée à l'IIIF Image API renverra un objet numérique de l'image. Afin d'accès aux métadonnées de l'image, on enverrait une requête d'une autre structure à la Presentaiton API, montrée dans la Figure 1b, qui renverrait des données en format JSON.

1.3 La mise en pratique

Dans le cadre du projet Artl@s en 2020, Caroline Corbières a développé un script pour importer à partir de l'IIIF Image API les pages d'un fac-similé numérique stocké sur les serveurs de la BnF. [2] Le projet *Gallic(orpor)a* a profité du travail de Corbières et l'a utilisé pour récupérer des fac-similés numériques ciblés par l'utilisatrice ou l'utilisateur du pipeline. En ajoutant une option au script (`-l`) je l'ai modifié afin de permettre qu'une utilisatrice ou utilisateur puisse limiter les nombres de pages récupérées. La limite évite le téléchargement de trop d'images ainsi qu'économise l'énergie dépensée. Ainsi, l'utilisatrice ou l'utilisateur peut tester la mise en œuvre des modèles HTR ou TAL sur un panel restreint des données.

2 L'application des modèles HTR

Le pipeline du projet *Gallic(orpor)a* visait à transcrire les ressources textuelles avec un modèle HTR qui convient bien au type du document, en rappelant qu'un modèle se spécialise dans une écriture particulière grâce à son entraînement. Cela exige donc que le pipeline a d'accès aux modèles entraînés et qu'il prend la décision automatiquement sur quel modèle convient auquel document. L'utilisateur ou l'utilisatrice doit donner des modèles au pipeline, pour qu'il puisse les appliquer aux données visuelles. Par contre, l'utilisateur ou l'utilisatrice ne doit pas forcément prendre la décision. Le pipeline saura quel modèle il doit appliquer en interrogeant les métadonnées du document et en recherchant sa langue et son siècle de création.

2.1 L'entraînement des modèles

Dans le cadre du projet *Gallic(orpor)a*, l'équipe a entraîné des nouveaux modèles généralisés, l'un pour les manuscrits et les incunables d'une écriture latine, l'autre pour les imprimés créées avant la révolution française et aussi d'une écriture latine. Ces modèles ont besoin des vérités de terrain produites lors du projet *Gallic(orpor)a*. Bien que le pipeline ne refasse pas la création de ces vérités de terrain, ne de l'entraînement des modèles, il compte de cet aspect du projet puisqu'il a besoin des modèles. Néanmoins, le pipeline peut s'adapter aux divers modèles et mettre en œuvre le modèle souhaités d'une 'utilisatrice ou d'un l'utilisateur. En s'adaptant aux modèles publiés ainsi qu'aux modèles toujours en cours, qu'une utilisatrice ou un utilisateur lui donne directement, le pipeline facilite les expériences scientifiques quant à l'HTR appliqué aux plusieurs corpus.

2. Simon Gabay et al. « Automating Artl@s – Extracting Data from Exhibition Catalogues ». In : *EADH 2021 - Second International Conference of the European Association for Digital Humanities*. Krasnoyarsk, Russia, sept. 2021. URL : `https://hal.archives-ouvertes.fr/hal-03331838` (visité le 23/08/2022).

2.1.1 La création des vérités de terrain

Lors de la première moitié de 2022, l'équipe du projet *Gallic(orpor)a* se composait des vacataires qui se chargeaient de la création des vérités de terrain pour les nouveaux modèles HTR. Les chefs du projet Simon Gabay et Ariane Pinche ont sélectionné un corpus de documents à transcrire. Les membres de l'équipe se spécialisaient dans certains types de document sur la liste, tels que les manuscrits et les incunables médiévaux. Un vacataire prendrait l'un des documents du corpus et saisir son identifiant ARK dans l'interface de transcription *eScriptorium*, dont je parle dans le chapitre **??**. L'interface imite ce que le pipeline du *Gallic(orpor)a* fait, ainsi que le script de Corbières, en téléchargeant les images IIIF du document.

L'interface *eScriptorium* facilite la segmentation et la transcription des pages. Les vacataires l'ont utilisaient pour segmenter les zones de la page et y mettre les étiquettes conformant au vocabulaire *SegmOnto*. Portant les étiquettes de *SegmOnto*, les transcriptions aident à entraîner les modèles avec un vocabulaire cohérent pour tout type de document, y compris les manuscrits et les imprimés. L'interface *eScriptorium* export les vérités de terrain dans un fichier XML ALTO avec des images télécharges en format JPEG ou PNG.

2.2 La sélection des modèles

Dans l'intérêt de maximiser l'automatisation du pipeline, la mise en œuvre d'un modèle s'effectue automatiquement, selon une analyse des métadonnées du document à traiter. Grâce au fait que les images sont diffusées par une API IIIF, le pipeline a d'accès à leurs métadonnées. Comme montre la Figure 1c, une requête au format de l'URI pour l'IIIF Presentation API renvoie des données qui portent sur la date de création ou d'apparition du document ainsi que la langue principale de son texte. Ces deux critères informent le pipeline du type de modèle à privilégier pour le document. Néanmoins, en l'absence du modèle parfait donné lors de l'installation de l'application *Gallic(orpor)a*, les paramètres du pipeline met en œuvre un modèle de segmentation et un modèle HTR désigné les modèles par défaut.

2.3 La sortie des modèles segmentation et HTR

Les modèles HTR produisent des fichiers ALTO, qui est un schème XML. La structure de données souhaitées dans le fichier final du pipeline est également de l'XML, qui se définie par l'imbrication des données. Mais au final, le pipeline produira un fichier TEI. La sortie des modèles HTR se conforme au schème ALTO et donc a besoin d'être transformée en TEI.

2.4 Le schème ALTO

La Bibliothèque nationale de France définit le schéma ALTO comme, « un schéma XML standardisé, qui permet de stocker les informations relatives à la

structure physique et au texte extrait par OCR ». [3] Les logiciels HTR exportent aussi leurs prédictions dans le format XML ALTO. Il y a les schémas XML qui conviennent bien à l'encodage du texte, tel que la TEI. Mais le liaison entre le texte et la mise en page, transcrite par un modèle de segmentation, est mieux établi par le schéma ALTO qui spécialise dans l'encodage de la structure physique d'une ressource textuelle.

Le schéma ALTO a été développé dans le cadre du projet METS (*Metadata Encoding and Transmission Schema*). Le dernier date de 2001. [4] Comme résume la Bibliothèque nationale de France,

> METS renseigne alors sur la structure logique de la page (nature sémantique des blocs de texte, par exemple titre, partie d'article, légende d'illustration, etc.), tandis qu'ALTO localise des contenants (blocs, lignes, etc.) sur la matrice de l'image de la page. [5]

Le schéma ALTO vise donc à renseigner sur la mise en page ainsi que sur le texte d'une page. Certains de ses éléments, tel que l'élément ALTO `<polygon>`, par exemple, précise les coordonnées d'une région ou d'une zone sur la page qu'avait prédite un modèle de segmentation. Cependant, certains attributs, tel que l'attribut `@CONTENT`, s'attribuent à certains parties de l'image du texte prédit. Par exemple, un schème ALTO encoderait une ligne de texte disant une phrase incomplète, *oyseaux lesq̃lz ie esperoye pren*, dans l'architecture XML qui ressemble à ce qui se voit dans la Figure 3. Tout cela veut dire que le schéma ALTO réussit à réunir les données structurelles de la mise en page et les données textuelles de la transcription prédite.

```
<TextLine ID="eSc_line_1ed06324"
    TAGREFS="LT825"
    BASELINE="1193 982 2263 969"
    HPOS="1184"
    VPOS="877"
    WIDTH="1079"
    HEIGHT="127">
    <Shape>
        <Polygon POINTS="1193 982 1184 903 1303 877 1307 877 2255 890
    2263 969 2263 995 1193 1004"/>
    </Shape>
    <String
        CONTENT="oyseaux˜lesqlz ie esperoye pren"
        HPOS="1184"
        VPOS="877"
        WIDTH="1079"
        HEIGHT="127">
    </String>
</TextLine>
```

3. Bertrand Caron. *Formats de Données Pour La Préservation à Long Terme : La Politique de La BnF*. Technical Report. Bibliothèque Nationale de France (Paris), oct. 2021. URL : https://hal-bnf.archives-ouvertes.fr/hal-03374030 (visité le 23/08/2022), p. 75.

4. *METS : Metadata Encoding and Transmission Standard*. BnF - Site institutionnel. URL : https://www.bnf.fr/fr/mets-metadata-encoding-and-transmission-standard (visité le 23/08/2022).

5. Caron, *Formats de Données Pour La Préservation à Long Terme*, p. 75.

FIGURE 3 – L'encodage d'une ligne de texte en ALTO

Le schéma XML ALTO peut se réaliser dans plusieurs formats, et pas uniquement celui qui se voit dans la Figure 3. Dans l'exemple de la Figure 3, les données textuelles sorties de la fonction prédictive du modèle HTR se trouvent dans l'attribut `@CONTENT` dans ce format. L'exemple montre un élément qui indique la région sur l'image source qu'occupe la ligne de texte, l'élément `<String>`. Son attribut `@CONTENT` porte sur le contenu textuel prédit sur cette ligne de texte.

Sinon, la sortie d'un logiciel HTR peut prendre les autres formats ALTO qui existent. L'un des défis pour le pipeline a été de le faire adapter aux divers formats d'ALTO qu'un logiciel HTR pourrait produire. Par exemple, le contenu textuel d'une ligne de texte sera encodé dans l'attribut `@CONTENT` de l'élément ALTO `<String>`, comme se voit dans la Figure 3, à la sortie de l'interface *eScriptorium*. Cependant, à la sortie de l'interface depuis la ligne de commande (CLI, ou *Command Line Interface*) de *Kraken*, le contenu textuel d'une ligne de texte est divisé entre les mots et les caractères reconnus dans la ligne. Par conséquent, le pipeline ne peut pas chercher le contenu de la ligne de texte dans l'attribut `@CONTENT` de l'élément `<String>`. Il doit reconstruire le contenu de la ligne de texte à partir de plusieurs éléments `<String>` qui représentent les mots dans une ligne de texte, au lieu de la ligne de texte elle-même.

3 Réunir la transcription et les métadonnées

Jusqu'ici, le pipeline a récupéré les images numériques en format JPEG ou PNG depuis l'IIIF Image API et les traité avec les modèles HTR en produisant des fichiers XML ALTO. Ensuite, le pipeline recherchera les métadonnées en plusieurs formats, y compris JSON, XML, et HTML, depuis l'internet. Avec une telle diversité de structures de données, un format robuste est exigé pour tout rassembler et mettre en ordre. Le format choisi pour parvenir à ce défi est l'XML TEI. Ce format se réalise dans un fichier XML, qui veut dire qu'il imbrique et les données dans une façon hiérarchisée. Mais contraire au schème XML ALTO, le schème TEI se spécialise dans l'édition numérique du texte. La communauté du TEI décrit l'objectif du projet ainsi :

> The Text Encoding Initiative (TEI) is a consortium which collectively develops and maintains a standard for the representation of texts in digital form. Its chief deliverable is a set of Guidelines which specify encoding methods for machine-readable texts, chiefly in the humanities, social sciences and linguistics. [6]

En plus de la représentation du texte, le TEI dispose aussi des éléments XML qui conviennent bien à la représentation des transcriptions, y compris leurs données topologiques portant sur la mise en page. Le TEI est donc un format idéal pour fusionner les transcriptions sorties des modèles HTR, les métadonnées

6. *TEI : Text Encoding Initiative*. URL : `https://tei-c.org/` (visité le 24/08/2022).

récupérées des sources en ligne, et le texte extrait des transcriptions. En plus, le TEI est un format très utilisé et beaucoup d'outils numériques s'y adaptent déjà.

3.1 L'extrait des données des fichiers ALTO

Les données sorties du traitement HTR sont du schème XML ALTO. L'un des objectifs du pipeline est de ne pas perdre aucune donnée produite par les modèles HTR, c'est-à-dire une donnée encodée dans le schéma ALTO. Il doit donc transformer la sortie des modèles en le format souhaité, le TEI. Il faut modéliser la transformation d'ALTO à TEI et la justifier puisqu'il y a plusieurs traductions possibles entre l'ALTO et le TEI.

Nous de l'équipe de *Gallic(orpor)a* avons conclu que l'élément XML TEI `<sourceDoc>` convient bien à l'encodage de toute donnée des modèles HTR portant sur le texte prédit et de la mise en page. Des autres chercheurs, tel que Hugo Scheithauer, Alix Chagué, et Laurent Romary, ont arrivés à la même conclusion.[7] Les guidelines de la Text Encoding Initiative définie l'élément `<sourceDoc>` ainsi :

> `<sourceDoc>` contains a transcription or other representation of a single source document potentially forming part of a dossier génétique or collection of sources.[8]

Les données topologiques et linguistiques sont balisés dans les éléments XML descendant du `<sourceDoc>`. Ces choix sont décrits et justifiés dans le chapitre **??**.

3.2 Le récupération des métadonnées

En plus de transformer les données des fichiers ALTO, le pipeline récupère les métadonnées sur le fac-similé numérique et le document source physique, ainsi que créer les métadonnées sur la ressource lexicographique elle-même. Toutes ses métadonnées sont encodées dans l'élément XML TEI `<teiHeader>`. Cet élément est essentiel au schème TEI, mais le pipeline réussit à construire ses descendants et les remplir avec les données si le fac-similé physique s'est trouvé dans la base de données Gallica. Sinon, le pipeline laisse vide la plupart des éléments obligatoires du `<teiHeader>`.

Avec l'identifiant ARK, qu'il prend du système de fichiers, le pipeline récupère les métadonnées du document source depuis les sources en ligne. Même pour les fac-similés hébergés par divers institutions, le pipeline récupère les métadonnées diffusées directement par l'API IIIF. Les métadonnées récupérés de l'API IIIF, géré par l'institution hôte du document numérique, sont liées—si possible—avec des autres sources de données. Dans l'exemple des documents

7. Hugo Scheithauer, Alix Chagué et Laurent Romary. « From eScriptorium to TEI Publisher ». In : *Brace Your Digital Scholarly Edition !* Berlin, France, nov. 2021. URL : `https://hal.inria.fr/hal-03538115` (visité le 23/08/2022).

8. *TEI element sourceDoc.* URL : `https://tei-c.org/release/doc/tei-p5-doc/en/html/ref-sourceDoc.html` (visité le 23/08/2022).

numériques de la base de données Gallica, le pipeline récupère les métadonnées quant au document source depuis l'API IIIF que la BnF met à disposition. Si cette requête a réussi, le pipeline va récupérer les données du catalogue général de la BnF en passant une requête à l'API *SRU* de la BnF qui veut dire, en anglais, le *Search/Retrieve via URL*. Pour terminer, le pipeline recherche encore des métadonnées dans le catalogue du Système Universitaire de Documentation (SUDOC) qui portent sur les institutions patrimoniales en France qui hébergent les documents sources.

3.3 La construction du fichier préliminaire TEI

Les données produites par les modèles HTR ainsi que les métadonnées récupérées depuis les sources en ligne sont réunies dans le fichier TEI. Les premiers dans l'élément `<sourceDoc>` et les dernières dans l'élément `<teiHeader>`. Ensuite, le pipeline commence à traiter les données intégrées au document. D'abord, il extrait les lignes de texte qui font partie des régions de la page considérées comme les principaux. Cette sélection du texte est comme une transcription puisqu'il ignore les en-têtes, les numéros de pages, etc. La transcription est ainsi encodée dans l'élément XML TEI `<body>`.

4 L'analyse linguistique et le fichier final

L'avant dernier étape du pipeline est d'analyser le texte. D'abord, les mots sont lemmatisés et reconnu selon leur nature, tel que nom ou verbe. La reconnaissance de la nature d'un mot se connaît par le terme anglais *part of speech* ou POS. La lemmatisation, un autre traitement lexical, divise un segment de texte en les parties et les indexer. Tout lexème pourrait ensuite être normalisé avec un modèle TAL qui transforme le mot prédit par le modèle HTR en sa version normalisée. Par exemple, un modèle TAL peut transformer le mot prédit *nostre* en le mot normalisé *nôtre*.[9] En fin, un modèle TAL NER (*Named-Entity Recognition*) peut encore traiter le texte pour reconnaître les entités nommées, tel qu'une personne ou un lieu. Le pipeline met en œuvre plusieurs modèles TAL afin d'analyser ses divers aspects linguistiques du texte.

4.1 L'ODD (One Document Does it all)

Un fichier TEI se soumettent aux règles qui assurent l'uniformité. Afin de mettre en pratique les traitements à l'échelle pour tout document produit par le pipeline, il faut que tout élément du TEI soit utilisé de la même manière. Pour parvenir à une telle régularisation, le TEI dispose d'un document qui applique des règles personnalisées au produit souhaité du pipeline. Ce document se connaît par son acronyme ODD, qui veut dire *One Document Does it all* ou

9. Rachel Bawden et al. « Automatic Normalisation of Early Modern French ». In : LREC 2022 - 13th Language Resources and Evaluation Conference. 20 juin 2022. doi : 10.5281/zenodo.5865428. url : https://hal.inria.fr/hal-03540226 (visité le 11/08/2022).

un fichier qui tout fait. Dans le cadre du stage, j'ai aidé à la rédaction d'un ODD qui explique en détail tout aspect du fichier TEI sorti du pipeline.

5 L'exploitation des données

En fin, les données encodées dans le fichier enrichi et final TEI peuvent être exploitées en tant qu'un fichier TEI ainsi que dans divers formats secondaires. Le logiciel TEI Publisher, par exemple, peut aisément publier un fichier TEI et permettre les utilisateurs de naviguer les données dans un visionneur de l'édition en ligne. Le fichier TEI peut aussi être exploité par les fichiers de transformation XSL. Il y a plusieurs formats secondaires qui pourraient servir à l'exploitation des données encodées dans le fichier TEI.

L'équipe a envisagé trois formats secondaires par lesquels les données produites par le pipeline peuvent être exploitées. L'un de ses formats est l'IIIF, le même qui a permis la construction du fichier TEI. Dans le fichier TEI, chaque attribut `@source` d'un élément `<zone>` descendant de l'élément `<sourceDoc>` contient un URI qui permet de visionner la partie de l'image concernée. Par exemple, si l'attribut `@source` descend d'un élément `<zone>` qui porte sur une ligne de texte, sa valeur donnerait dans un browser ou dans un visionneur uniquement la partie de l'image source qui contient cette ligne de texte. Ainsi les données du fichier final TEI peuvent être exploitées afin de visionner les bloques de textes, lignes de textes, les mots, et les caractères transcrits. Les deux autres formats secondaires profitent plutôt des métadonnées du fichier TEI ; ils sont le DTS (Distributed Text Services) et le RDF (Resource Description Framework).