ÉCOLE NATIONALE DES CHARTES
UNIVERSITÉ PARIS, SCIENCES & LETTRES

---

**Kelly Christensen**
*diplômée de doctorat musicologie*

# D'ALTO à TEI

## Modélisation de transcriptions automatiques pour une pré-éditorialisant des textes

Mémoire pour le diplôme de master
« Technologies numériques appliquées à l'histoire »

2022

# Résumé

Quand des modèles OCR et HTR extraient les données d'une ressource textuelle numérisée, les informations relatives à la structure physique de l'image risquent de se perdre. Un schéma XML standardisé qui s'appelle ALTO a été créé afin de conserver et structurer ces données non-textuelles et géometriques en les tenant en relation avec le contenu textuel. La plupart des modèles OCR et HTR compte sur ce schéma. Cependant ALTO ne convient pas bien à l'édition numérique ni aux traitements automatique du langage. Les éditeurs et les chercheurs en lettres attendent un schéma XML plus courant dans le monde des humanités numériques : la TEI. Il faut donc un mapping pour transformer un fichier ALTO en fichier TEI sans perdre aucune donnée lors du processus. Cette transformation automatisée permet à conserver les données particulières au schéma ALTO, telles que celles sur la segmentation et sur la structure physique du document numérisé, ainsi qu'à exploiter le contenu textuel de la ressource textuelle. La flexibilité de la TEI et son usage très répandu rendent le schéma idéal pour mieux valoriser les données produites par les modèles OCR et HTR.

Dans le cadre du stage pour obtenir le diplôme de Master 2 « Technologies numériques appliquées à l'histoire », ce mémoire porte sur la modélisation de la transformation de ALTO en TEI. Cette modélisation a été réalisée dans le cadre du projet *Gallic(orpor)a*, financé par la BnF lors d'un stage qui a eu lieu au sein du laboratoire Automatic Language Modelling and Analysis & Computational Humanities entre avril et juillet 2022.

**Mots-clés :** HTR, OCR, ALTO, TEI, TAL, édition numérique.

**Informations bibliographiques :** Kelly Christensen, *D'ALTO à TEI, modélisation de transcriptions automatiques pour une pré-éditorialisant des textes*, mémoire de master « Technologies numériques appliquées à l'histoire », dir. Ségolène Albouy, École nationale des chartes, 2022.

# Remerciements

Mes remerciements vont tout d'abord à...

# Bibliographie


*16 Linking, Segmentation, and Alignment - The TEI Guidelines*. URL : https://tei-c.org/release/doc/tei-p5-doc/en/html/SA.html#SASOstdf (visité le 25/08/2022).

*About – TEI : Text Encoding Initiative*. URL : https://tei-c.org/about/ (visité le 25/08/2022).

Alliance, The ARK. *Community*. ARK Alliance. URL : https://arks.org/community/ (visité le 23/08/2022).

Bartz, Alexandre et Juliette Janes. *Annotator*. E-ditiones. URL : https://github.com/e-ditiones/Annotator (visité le 07/09/2022).

Bartz, Alexandre et al. « Expanding the Content Model of annotationBlock ». In : *Next Gen TEI, 2021 - TEI Conference and Members' Meeting*. Virtual, United States, oct. 2021. URL : https://hal.archives-ouvertes.fr/hal-03380805 (visité le 07/09/2022).

Bawden, Rachel. *ModFr-Normalisation*. URL : https://github.com/rbawden/ModFr-Norm (visité le 07/09/2022).

Bawden, Rachel 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).

Berchmans, Deepa et S S Kumar. « Optical Character Recognition : An Overview and an Insight ». In : *2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT)*. 2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT). Juill. 2014, p. 1361-1365. DOI : 10.1109/ICCICCT.2014.6993174.

Bibliothèque nationale de France. *Rapport d'activité 2021 de la Bibliothèque nationale de France*. Paris, France, 1er juill. 2022. URL : https://www.bnf.fr/fr/bnf-rapport-dactivite-2021 (visité le 09/08/2022).

Bobichon, Philippe. « Le Lexicon : Mise En Page et Mise En Texte Des Manuscrits Hébreux, Grecs, Latins, Romans et Arabes ». In : (2009), p. 81. URL : https://cel.archives-ouvertes.fr/cel-00377671 (visité le 25/07/2022).

Bobichon, Philippe. *Caviarder*. In : *Codicologia*. Institut de recherche et d'histoire des textes, 2011. URL : http://codicologia.irht.cnrs.fr/theme/liste_theme/413#tr-868 (visité le 25/07/2022).

Camps, Jean-Baptiste et al. « Corpus and Models for Lemmatisation and POS-tagging of Classical French Theatre ». In : *Journal of Data Mining & Digital Humanities* 2021 (Digital humanities in... 14 fév. 2021), p. 6485. ISSN : 2416-5999. DOI : 10.46298/jdmdh.6485. arXiv : 2005.07505 [cs]. URL : http://arxiv.org/abs/2005.07505 (visité le 09/08/2022).

Carlin, Marie et Arnaud Laborderie. « Le BnF DataLab, Un Service Aux Chercheurs En Humanités Numériques ». In : *Humanités numériques* 4 (déc. 2021). URL : https://hal-bnf.archives-ouvertes.fr/hal-03285816 (visité le 11/08/2022).

Caron, Bertrand. *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).

Chagué, Alix. *LECTAUREP Contemporary French Model (Administration)*. Zenodo, 12 mai 2022. URL : https://zenodo.org/record/6542744 (visité le 12/08/2022).

Chagué, Alix et Thibault Clérice. « Sharing HTR Datasets with Standardized Metadata : The HTR-United Initiative ». In : Documents Anciens et Reconnaissance Automatique Des Écritures Manuscrites. 23 juin 2022. URL : https://hal.inria.fr/hal-03703989 (visité le 12/08/2022).

Chagué, Alix, Thibault Clérice et Laurent Romary. « HTR-United : Mutualisons La Vérité de Terrain ! » In : *DHNord2021 - Publier, Partager, Réutiliser Les Données de La Recherche : Les Data Papers et Leurs Enjeux*. Lille, France : MESHS, nov. 2021. URL : https://hal.archives-ouvertes.fr/hal-03398740 (visité le 10/08/2022).

Chagué, Alix et al. *HTR-United/Htr-United : V0.1.28*. Zenodo, 10 août 2022. DOI : 10.5281/zenodo.6979746. URL : https://zenodo.org/record/6979746 (visité le 12/08/2022).

Clérice, Thibault. *Pie Extended, an Extension for Pie with Pre-Processing and Post-Processing*. Zenodo, juin 2020. DOI : 10.5281/zenodo.6534764. URL : https://zenodo.org/record/6534764 (visité le 07/09/2022).

— *YALTAi : Segmonto Manuscript and Early Printed Book Dataset*. 10 juill. 2022. DOI : 10.5281/zenodo.6814770. URL : https://zenodo.org/record/6814770 (visité le 12/08/2022).

Clérice, Thibault et Ariane Pinche. *HTRVX, HTR Validation with XSD*. Version 0.0.1. Sept. 2021. DOI : 10.5281/zenodo.5359963. URL : https://github.com/HTR-United/HTRVX (visité le 12/08/2022).

Coquenet, Denis, Clément Chatelain et Thierry Paquet. « Handwritten Text Recognition : From Isolated Text Lines to Whole Documents ». In : *ORASIS 2021*. Saint

Ferréol, France : Centre National de la Recherche Scientifique [CNRS], sept. 2021. URL : https://hal.archives-ouvertes.fr/hal-03339648 (visité le 04/08/2022).

*DigiPal : Digital Resource and Database of Palaeography, Manuscripts and Diplomatic.* London. URL : http://www.digipal.eu/.

GABAY, Simon. *E-Ditiones, 17th c. French Sources*. Nov. 2018. URL : https://hal.archives-ouvertes.fr/hal-02388415 (visité le 10/08/2022).

— *FreEM-corpora/FreEMnorm : FreEM Norm Parallel Corpus*. Zenodo, 17 jan. 2022. DOI : 10.5281/zenodo.5865428. URL : https://zenodo.org/record/5865428 (visité le 11/08/2022).

GABAY, Simon, Jean-Baptiste CAMPS et Ariane PINCHE. « SegmOnto ». In : *Création de Modèle(s) HTR Pour Les Documents Médiévaux En Ancien Français et Moyen Français Entre Le Xe-XIVe Siècle*. Paris, France : Ecole nationale des chartes | PSL, nov. 2021. URL : https://hal.archives-ouvertes.fr/hal-03481089 (visité le 25/07/2022).

GABAY, Simon, Thibault CLÉRICE et Christian REUL. *OCR17 : Ground Truth and Models for 17th c. French Prints (and Hopefully More)*. Mai 2020. URL : https://hal.archives-ouvertes.fr/hal-02577236 (visité le 12/08/2022).

GABAY, Simon et al. « Standardizing Linguistic Data : Method and Tools for Annotating (Pre-Orthographic) French ». In : *Proceedings of the 2nd International Digital Tools & Uses Congress (DTUC '20)*. Hammamet, Tunisia, oct. 2020. DOI : 10.1145/3423603.3423996. URL : https://hal.archives-ouvertes.fr/hal-03018381 (visité le 12/08/2022).

GABAY, Simon 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).

GABAY, Simon et al. « SegmOnto : Common Vocabulary and Practices for Analysing the Layout of Manuscripts (and More) ». In : *1st International Workshop on Computational Paleography (IWCP@ICDAR 2021)*. Lausanne, Switzerland, sept. 2021. URL : https://hal.archives-ouvertes.fr/hal-03336528 (visité le 09/08/2022).

GABAY, Simon et al. « From FreEM to D'AlemBERT ». In : *Proceedings of the 13th Language Resources and Evaluation Conference*. Marseille, France : European Language Resources Association, juin 2022. URL : https://hal.inria.fr/hal-03596653 (visité le 10/08/2022).

GABAY, Simon et al. « Le Projet FREEM : Ressources, Outils et Enjeux Pour l'étude Du Français d'Ancien Régime ». In : *TALN 2022 - Traitement Automatique Des Langues Naturelles*. Sous la dir. d'Yannick ESTÈVE et al. Avignon, France : ATALA, juin 2022, p. 154-165. URL : https://hal.archives-ouvertes.fr/hal-03701524 (visité le 10/08/2022).

GACEK, Adam. *The Arabic Manuscript Tradition : A Glossary of Technical Terms and Bibliography*. Handbook of Oriental Studies 1, The Near and Middle East. Leiden : Brill, 2001. 269 p. ISBN : ISBN 90-04-12061-0.

« Glossaire codicologique français-arabe ». In : *Gazette du livre médiéval* 40.1 (2002), p. 79-80. URL : https://www.persee.fr/doc/galim_0753-5015_2002_num_40_1_1563 (visité le 25/07/2022).

GOODRICH, Gregory. « Kurzweil Reading Machine : A Partial Evaluation of Its Optical Character Recognition Error Rate ». In : *Journal of Visual Impairment and Blindness* (12 jan. 1979).

GOVINDAN, V. K. et A. P. SHIVAPRASAD. « Character Recognition — A Review ». In : *Pattern Recognition* 23.7 (1990), p. 671. ISSN : 0031-3203. URL : https://www.academia.edu/6986960/Character_recognition_A_review (visité le 05/08/2022).

GUIBON, Gaël et al. « Parsing Poorly Standardized Language Dependency on Old French ». In : *Thirteenth International Workshop on Treebanks and Linguistic Theories (TLT13)*. Sous la dir. de V. HENRICH et al. Proceedings of the Thirteenth International Workshop on Treebanks and Linguistic Theories (TLT13). Tübingen, Germany, déc. 2014, p. 51-61. URL : https://hal.archives-ouvertes.fr/hal-01250959 (visité le 10/08/2022).

IMPEDOVO, Sebastiano. « More than Twenty Years of Advancements on Frontiers in Handwriting Recognition ». In : *Pattern Recognition*. Handwriting Recognition and Other PR Applications 47.3 (1er mars 2014), p. 916-928. ISSN : 0031-3203. DOI : 10.1016/j.patcog.2013.05.027. URL : https://www.sciencedirect.com/science/article/pii/S0031320313002513 (visité le 29/08/2022).

JENTSCH, Patrick et Stephan PORADA. « From Text to Data Digitization, Text Analysis and Corpus Linguistics ». In : *Digital Methods in the Humanities : Challenges, Ideas, Perspectives*. Sous la dir. de Silke SCHWANDT. Bielefeld University Press, 2021.

KIESSLING, Benjamin. « Kraken - an Universal Text Recognizer for the Humanities ». In : ADHO 2019 - Utrecht. 2019. URL : https://dh-abstracts.library.cmu.edu/works/9912 (visité le 10/08/2022).

LASSNER, David et al. « Publishing an OCR Ground Truth Data Set for Reuse in an Unclear Copyright Setting. Two Case Studies with Legal and Technical Solutions to Enable a Collective OCR Ground Truth Data Set Effort ». Version 1.0. In : *Fabrikation von Erkenntnis – Experimente in den Digital Humanities. Hg. von Manuel Burghardt* Lisa Dieckmann (2021). Avec la coll. d'Herzog August BIBLIOTHEK, 5). DOI : 10.17175/SB005_006. URL : https://zfdg.de/sb005_006 (visité le 10/08/2022).

MANIACI, Marilena. *Terminologia Des Libro Manoscritto*. Addenda 3. Rome : Istituto centrale per la patologia del libro, 1996.

*Manuel UNIMARC : format bibliographique*. Transition bibliographique - Programme national. URL : https://www.transition-bibliographique.fr/unimarc/manuel-unimarc-format-bibliographique/ (visité le 27/08/2022).

MARTIN, Louis et al. « CamemBERT : A Tasty French Language Model ». In : *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. ACL 2020. Online : Association for Computational Linguistics, juill. 2020, p. 7203-7219. DOI : 10.18653/v1/2020.acl-main.645. URL : https://aclanthology.org/2020.acl-main.645 (visité le 11/08/2022).

*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).

MISA, Thomas J. *Gender Codes : Why Women Are Leaving Computing*. John Wiley & Sons, 14 sept. 2011. 326 p. ISBN : 978-1-118-03513-9. Google Books : EjDYh_KHls8C.

MUEHLBERGER, Guenter et al. « Transforming Scholarship in the Archives through Handwritten Text Recognition : Transkribus as a Case Study ». In : *Journal of Documentation* 75.5 (9 sept. 2019), p. 954-976. ISSN : 0022-0418. DOI : 10.1108/JD-07-2018-0114. URL : https://www.emerald.com/insight/content/doi/10.1108/JD-07-2018-0114/full/html (visité le 04/08/2022).

MUZERELLE, Denis. *Caviarder*. In : *Codicologia*. Institut de recherche et d'histoire des textes, 2011. URL : http://codicologia.irht.cnrs.fr/theme/liste_theme/413#tr-868 (visité le 25/07/2022).

MUZERELLE, Dennis. *Vocabulaire Codicologique : Répertoire Méthodique Des Termes Français Relatifs Aux Manuscrits*. Rubricae 1. Paris : Éd. Cemi, 1985.

*Numpy : NumPy Is the Fundamental Package for Array Computing with Python*. Version 1.23.1. URL : https://www.numpy.org (visité le 04/08/2022).

ORTIZ SUÁREZ, Pedro Javier, Benoît SAGOT et Laurent ROMARY. « Asynchronous Pipeline for Processing Huge Corpora on Medium to Low Resource Infrastructures ». In : *7th Workshop on the Challenges in the Management of Large Corpora (CMLC-7)*. Sous la dir. de Piotr BAŃSKI et al. Cardiff, United Kingdom : Leibniz-Institut für Deutsche Sprache, juill. 2019. DOI : 10.14618/IDS-PUB-9021. URL : https://hal.inria.fr/hal-02148693 (visité le 07/09/2022).

OSTOS, Pilar, M. Luisa PARDO et Elena E. RODRÍGUEZ. *Vocabulario de codicologÍa : Versión Española Revisada y Aumentada Del Vocabulaire Codicologique*. Instrumenta Bibliologica. Madrid : Arco/Libros, 1997.

OTSU, Nobuyuki. « A Threshold Selection Method from Gray-Level Histograms ». In : *IEEE Transactions on Systems, Man, and Cybernetics* 9.1 (jan. 1979), p. 62-66.

PELLET, Aurélien et Marie PUREN. « Le Projet AGODA. Océrisation Des Débats Parlementaires Français de La Troisième République : Problèmes, Défis et Perspectives ». In : Séminaire OMNSH-Epitech : Le Numérique Au Service Des Sciences Humaines

et Sociales. Le Kremlin-Bicêtre, France, avr. 2022. URL : https://hal.archives-ouvertes.fr/hal-03651146 (visité le 29/08/2022).

PINCHE, Ariane. « HTR model Cremma Medieval ». In : (21 juin 2022). DOI : 10.5281/zenodo.6669508. URL : https://zenodo.org/record/6669508 (visité le 10/08/2022).

PINCHE, Ariane et Jean-Baptiste CAMPS. « CremmaLab Project : Transcription Guidelines and HTR Models for French Medieval Manuscripts ». In : *Colloque "Documents Anciens et Reconnaissance Automatique Des Écritures Manuscrites"*. Paris, France, juin 2022. URL : https://hal.archives-ouvertes.fr/hal-03716526 (visité le 10/08/2022).

PINCHE, Ariane et Thibault CLÉRICE. *HTR-United/Cremma-Medieval : Cortado 2.0.0*. Zenodo, 11 juill. 2022. DOI : 10.5281/zenodo.6818057. URL : https://zenodo.org/record/6818057 (visité le 12/08/2022).

PROJECT, CatCor. *Letter 02633 : To Frederick II (the Great), 21 July 1744*. Sous la dir. d'Andrew KAHN et RUBIN-DETLEV. 2021. URL : https://catcor.seh.ox.ac.uk/id/letter-02633.

REGNAULT, Mathilde, Sophie PRÉVOST et Éric Villemonte de LA CLERGERIE. « Challenges of Language Change and Variation : Towards an Extended Treebank of Medieval French ». In : *TLT 2019 - 18th International Workshop on Treebanks and Linguistic Theories*. Paris, France, août 2019. URL : https://hal.inria.fr/hal-02272560 (visité le 10/08/2022).

SCHEITHAUER, Hugo, 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).

SCHNEIDER, Ben R. « The Production of Machine-Readable Text : Some of the Variables ». In : *Computers and the Humanities* 6.1 (sept. 1971), p. 39-47. ISSN : 0010-4817, 1572-8412. DOI : 10.1007/BF02402324. URL : http://link.springer.com/10.1007/BF02402324 (visité le 02/08/2022).

SIDDIQI, Imran-Ahmed, Florence CLOPPET et Nicole VINCENT. « Writing Property Descriptors : A Proposal for Typological Groupings ». In : *Gazette du livre médiéval* 56.1 (2011), p. 42-57. DOI : 10.3406/galim.2011.1981. URL : https://www.persee.fr/doc/galim_0753-5015_2011_num_56_1_1981 (visité le 02/08/2022).

*SRU : Search/Retrieval via URL – SRU, CQL and ZeeRex (Standards, Library of Congress)*. URL : https://www.loc.gov/standards/sru/ (visité le 02/09/2022).

STEHNO, Birgit, Alexander EGGER et Gregor RETTI. « METAe–Automated Encoding of Digitized Texts ». In : *Literary and Linguistic Computing* 18.1 (1er avr. 2003), p. 77-88. ISSN : 0268-1145, 1477-4615. DOI : 10.1093/llc/18.1.77. URL : https://academic.oup.com/dsh/article-lookup/doi/10.1093/llc/18.1.77 (visité le 24/08/2022).

STEIN, Achim et Sophie PRÉVOST. *Syntactic Annotation of Medieval Texts : The Syntactic Reference Corpus of Medieval French (SRCMF)*. Narr Verlag, 2013, p. 275. ISBN : 978-3-8233-6760-4. URL : https://halshs.archives-ouvertes.fr/halshs-01122079 (visité le 10/08/2022).

— *Syntactic Reference Corpus of Medieval French (SRCMF)*. Stuttgart : ILR University of Stuttgart, 2013. ISBN : 899-492-963-833-3. URL : http://srcmf.org.

*TEI element facsimile*. URL : https://tei-c.org/release/doc/tei-p5-doc/en/html/ref-facsimile.html (visité le 03/09/2022).

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

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

TFIBEL, Florence. *RE : Tr : [Gallic(Orpor)a] Question Sur l'Unimarc Du Catalogue Généra*. E-mail. 13 juill. 2022.

*The Bodelian First Folio : Digital Facsimile of the First Folio of Shakespeare's Plays*. URL : http://firstfolio.bodleian.ox.ac.uk/ (visité le 27/08/2022).

TRUAN, Naomi et Laurent ROMARY. « Building, Encoding, and Annotating a Corpus of Parliamentary Debates in TEI XML : A Cross-Linguistic Account ». In : *Journal of the Text Encoding Initiative* Issue 14 (Issue 14 17 mars 2021). ISSN : 2162-5603. DOI : 10.4000/jtei.4164. URL : https://journals.openedition.org/jtei/4164#tocto2n4 (visité le 26/08/2022).

TURING, A. M. « Computing Machinery and Intelligence ». In : *Mind* LIX.236 (1er oct. 1950), p. 433-460. ISSN : 0026-4423. DOI : 10.1093/mind/LIX.236.433. URL : https://doi.org/10.1093/mind/LIX.236.433 (visité le 04/08/2022).

ZARRI, Gian Piero. « Quelques aspects techniques de l'exploitation informatique des documents textuels : saisie des données et problèmes de sortie ». In : *Publications de l'École Française de Rome* 31.1 (1977), p. 399-413. URL : https://www.persee.fr/doc/efr_0000-0000_1977_act_31_1_2286 (visité le 01/08/2022).

ref

# Glossaire

**CREMMALab** Consortium pour la reconnaissance d''écriture manuscrite des matériaux anciens. 9

**Handwritten Text Recognition** La reconnaissance du texte écrit sur une image numérique. 21

**HTR-United** HTR-United is a catalog and an ecosystem for sharing and finding ground truth for optical character or handwritten text recognition (OCR/HTR). 7–10

**IIIF Image API** Un service de web qui renvoie une image suite à une requête standardisée HTTP(S). L'URI peut préciser la région, la taille, la rotation, la qualité, les caractéristiques, et le format de l'image demandée.. 41, 42, 45, 83, 94

**Inria** Institut national de recherche en sciences et technologies du numérique. 3–5, 10

**International Image Interoperability Framework** Normes internationales de l'exploitation des images numériques et de leurs métadonnées par API. 40

**One Document Does it all** Un fichier XML TEI qui précise les règles d'un schème TEI personnalisé.. 48

**Optical Character Recognition** La reconnaissance des polices du texte sur une image numérique. 21

**UNIMARC** Une référence pour l'échange de données en format XML. 63, 64, 67, 70, 76, 77, 79–81, 84, 86, 90, 120, 127

**École nationale des chartes** Grande école bla bla bla. 5

# Acronymes

**ALMAnaCH** Automatic Language Modelling and Analysis & Computational Humanities. 4, 5, 8

**ALTO** Analyzed Layout and Text Object. i, 13, 43–47, 49–51, 53, 54, 56, 57, 65, 75, 77, 78, 82, 84, 86, 88, 91–105, 107, 120, 126

**API** Application Programming Interface. 40–43, 47, 64, 75–84, 95, 96, 120

**ARK** Archival Resource Key. 39–41, 43, 47, 69, 76–80, 94, 96, 112, 120, 121, 126

**BnF** Bibliothèque nationale de France. i, 3–5, 40–42, 44, 47, 59, 62, 64, 66, 67, 69, 70, 75–84, 86, 87, 90, 94

**CREMMA** Consortium Reconnaissance d'Écriture Manuscrite des Matériaux Anciens. 9, 10

**CSV** Comma Separated Values. 112

**DTS** Distributed Text Services. 39, 48

**ENC** École nationale des chartes. 5

**HTML** HyperText Markup Language. 45, 81, 84, 90

**HTR** Handwritten Text Recognition. i, 3, 5, 6, 8–10, 12, 13, 21, 23–29, 31–34, 39, 42–47, 50, 52, 57, 59, 65, 67, 70, 75–77, 82, 84, 88, 91–93, 95–97, 99, 100, 107, 108, 111

**IIIF** International Image Interoperability Framework. 39–41, 43, 47, 48, 64, 67, 70, 76–80, 82–84, 86, 87, 90, 94–96, 120, 126, 127

**Inria** Institut national de recherche en sciences et technologies du numérique. 4

**JSON** JavaScript Object Notation. 42, 45, 64, 76, 84, 90

**METS** Metadata Encoding and Transmission Standard. 49, 50

**OCR** Optical Character Recognition. i, 9, 21–23, 25, 27, 29–32, 44, 49, 50, 52, 57

**ODD** One Document Does it all. 48

**RDF** Resource Description Framework. 39, 48

**SRU** Search/Retrieve via URL. 64, 76, 77, 79–81, 84

**SUDOC** Système Universitaire de Documentation. 76, 77, 81, 82, 84, 90, 120, 127

**TAL** Traitement automatique des langues. 3–5, 10–12, 14, 33, 42, 47, 48, 56, 107, 110–112, 121

**TEI** Text Encoding Initiative. i, 13, 14, 39, 44, 46–48, 54, 56, 57, 59–64, 70, 75, 76, 79, 81, 82, 84, 91–105, 107–109, 113, 120

**TSV** Tab Separated Values. 112

**XML** eXtensible Markup Language. 13, 43–47, 49–51, 54, 60, 63, 75, 76, 79, 82, 84, 90–92, 101, 112

**YAML** Yet Another Markup Language. 76, 82–84, 90

# Introduction

# Première partie

# Présentation du projet

# Chapitre 1

# Le rêve du projet *Gallic(orpor)a*

Le projet *Gallic(orpor)a* s'est développé à partir de plusieurs projets précédents et il tire parti de divers domaines de recherche. Ses créateurs, en mettant en valeur leurs propres connaissances, ont visé à assembler un pipeline qui peut traiter tout document dans la base de données Gallica de la Bibliothèque nationale de France (BnF). Les chercheurs spécialisés en *l'Handwritten Text Recognition* (HTR), en le Traitement automatique des langues (TAL), en l'histoire, en la littérature, en la lexicographie et en la stylométrie se sont rassemblés pour réaliser ce pipeline. Le pipeline visait à prédire et analyser du ancien français et du français de l'Ancien Régime, ainsi que les manuscrits et les imprimés, à partir des pages numérisées des documents créés entre 1400 et la révolution française. Cependant, le vrai rêve du projet était de produire un prototype qui servirait d'exemple et pourrait être élaboré dans le but de traiter vraiment tout document source numérisé.

Les ambitions du *Gallic(orpor)a* se sont rendu possibles grâce aux recherches de plusieurs chercheurs et ingénieurs, tel que Laurent Romary, Philippe Gambette, Thibault Clérice, Pedro Suarez Ortiz, Claire Jahan, Caroline Corbières, et Alexandre Bartz. Mais les principaux qui se chargeaient de la surveillance du projet *Gallic(orpor)a* lors de mon stage en 2022 étaient Jean-Baptiste Camps, Simon Gabay, et Ariane Pinche, qui ont développé des modèles HTR pour extraire du texte des document numériques dans la base de données Gallica. Chez Inria, en tant que stagiaire, j'ai aussi travaillé en collaboration avec Benoît Sagot et Rachel Bawden, qui ont développé des outils d'analyse linguistique du texte extrait. Tous ensemble, ces chercheurs de divers spécialités ont contribué leurs connaissances pour produire un processus du traitement polyvalent.

## 1.1 Le contexte du projet

### Bibliothèque nationale de France et le Data Lab

Le Data Lab s'est mis en place au sein de la Bibliothèque nationale de France (BnF) en 2021.[1] Lors de sa première année, le Data Lab a lancé son premier appel aux projets qui mettent en valeur les fonds et les ressources de l'institution phare patrimoniale. Le projet *Gallic(orpor)a* faisait partie des premiers projets acceptés en 2021, à côté des projets *AUREJ* (Accès Unifié aux REssources de la Jouabilité), *GALLICAENV*, *BUZZ-F*, et *AGODA* (Analyse sémantique et Graphes relationnels pour l'Ouverture et l'étude des Débats à l'Assemblée nationale).[2] Ayant sa candidature retenu, *Gallic(orpora)* profitait du financement du Data Lab de la BnF. La plupart du travail sur le projet a eu lieu pendant la première moitié de 2022, suite au mis en place du stage et des vacations par Ariane Pinche, Simon Gabay, et Benoît Sagot.

### Inria et l'équipe ALMAnaCH

Inria est l'Institut national de recherche en sciences et technologies du numérique et il compte plusieurs branches dans le monde. La branche parisienne encadre l'équipe ALMAnaCH dont le acronyme veut dire *Automatic Language Modelling and Analysis & Computational Humanities*. Au sein d'ALMAnaCH s'est développé le meilleur modèle TAL pour la langue française, CamemBERT.[3] L'équipe ALMAnaCH encadre les chercheurs, les ingénieurs, les doctorants, et les stagiaires attachés aux projets concernés soit par le traitement automatique des langues, soit par les humanités numériques. L'acronyme du nom fait référence à ces deux pôles de recherche : *Automatic Language Modelling and Analysis* est le traitement automatique des langues, et le *Computational Humanities* est l'humanités numériques. Le projet *Gallic(orpor)a* se situait entre les deux, impliquant l'extraction des données et l'édition des documents historiques ainsi que l'analyse linguistique du texte extrait.

Le directeur de recherches d'ALMAnaCH est Benoît Sagot, qui s'est chargé de l'encadrement du stage du projet *Gallic(orpor)a*. En tant que stagiaire, je faisais partie de l'équipe entre début avril et fin juillet 2022. Pendant le stage, Rachel Bawden a animé un groupe de lecture hebdomadaire et des séminaires mensuelles dont j'ai profité dans

1. Marie Carlin et Arnaud Laborderie. « Le BnF DataLab, Un Service Aux Chercheurs En Humanités Numériques ». In : *Humanités numériques* 4 (déc. 2021). URL : `https://hal-bnf.archives-ouvertes.fr/hal-03285816` (visité le 11/08/2022).

2. Bibliothèque nationale de France. *Rapport d'activité 2021 de la Bibliothèque nationale de France*. Paris, France, 1er juill. 2022. URL : `https://www.bnf.fr/fr/bnf-rapport-dactivite-2021` (visité le 09/08/2022), p. 123.

3. Louis Martin et al. « CamemBERT : A Tasty French Language Model ». In : *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. ACL 2020. Online : Association for Computational Linguistics, juill. 2020, p. 7203-7219. DOI : `10.18653/v1/2020.acl-main.645`. URL : `https://aclanthology.org/2020.acl-main.645` (visité le 11/08/2022).

l'intérêt de me tenir au courant sur les nouvelles recherches et les nouveaux enjeux du TAL. Elle a aussi développé un modèle TAL pour le projet *Gallic(orpor)a* et m'a instruit dans sa mise en œuvre. [4] Disposée d'un bureau, j'ai travaillé en présentiel quatre jours par semaine, passant un jour toutes les semaines à l'École nationale des chartes pour travailler à côté de l'une des chefs du projet, Ariane Pinche. Chez Inria, j'ai profité de l'expertise de mes collègues de l'équipe ALMAnaCH, en particulier Alix Chagué et Hugo Scheithauer. L'équipe entière de *Gallic(orpor)a* a aussi profité des serveurs d'Inria, qui prennaient en charge une partie de la puissance de calcul et du stockage de données pour l'interface graphique HTR *eScriptorium*.

### École nationale des chartes et l'université de Genève

En tant qu'une école, contrairement à une équipe de recherche comme ALMAnaCH, les rôles de l'École nationale des chartes et l'université de Genève dans le projet *Gallic(orpor)a* concernés l'encadrement des chercheurs qui y ont contribué leurs connaissances. L'École nationale des chartes (ENC) a aussi donné un lieu de travail, dont j'ai profité un jour par semaine. Ariane Pinche, qui était post-doctorante à l'École nationale des chartes, et Simon Gabay, maître-assistant à l'université de Genève, ils ont géré la mise en place du stage et des vacations que la bourse du Data Lab de la BnF a financés pour 2022. L'ENC et l'université de Genève les ont soutenu lors de l'encadrement des vacations et du stage.

Gabay, Pinche, et deux autres chercheurs qui étaient attachés à l'École nationale des chartes pendant le stage, Jean-Baptiste Camps et Thibault Clérice, ont tous contribué au projet *Gallic(orpor)a*. Gabay et Pinche se sont occupés de l'harmonisation des vérités de terrain produites par l'équipe en reliant toute transcription que les vacataires ont faite dans l'interface graphique *eScriptorium*. Pinche et Clérice ont commencé à utilisé les vérités de terrain des documents médiévaux en entraînant des nouveaux modèles de l'HTR et de la segmentation. [5] Par rapport à la segmentation, Jean-Baptiste Camps, Pinche, et Gabay ont développé le syntaxe *SegmOnto* qui servait à harmoniser les vérités de terrain produites pour tout document dans le corpus d'entraînement, y compris les manuscrits et les imprimés. [6] Bien que chaque chercheur ait ses spécialités, ils ont tous collaboré et la

4. 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) ; Simon Gabay. *FreEMcorpora/FreEMnorm : FreEM Norm Parallel Corpus*. Zenodo, 17 jan. 2022. DOI : 10.5281/zenodo.5865428. URL : https://zenodo.org/record/5865428 (visité le 11/08/2022).

5. Thibault Clérice. *YALTAi : Segmonto Manuscript and Early Printed Book Dataset*. 10 juill. 2022. DOI : 10.5281/zenodo.6814770. URL : https://zenodo.org/record/6814770 (visité le 12/08/2022).

6. Simon Gabay et al. « SegmOnto : Common Vocabulary and Practices for Analysing the Layout of Manuscripts (and More) ». In : *1st International Workshop on Computational Paleography (IWCP@ICDAR 2021)*. Lausanne, Switzerland, sept. 2021. URL : https://hal.archives-ouvertes.fr/hal-03336528 (visité le 09/08/2022).

division des aspects du projet n'étaient pas aussi fermes qu'ils n'ont pas profité des idées de l'un et de l'autre.

## 1.2 La portée des données d'entraînement

Le projet *Gallic(orpor)a* visait à développer et mettre en pratique les modèles HTR pouvant traiter tout document français dans la base de données Gallica crée entre le XIVe siècle et la révolution française. Un corpus des documents sources numérisés ainsi qu'une sélection de leurs pages à transcrire étaient choisis comme les données d'entraînement d'un tel modèle HTR. Les membres de ce corpus sont détaillés dans le section A.1 de l'appendice. Le choix se faisait en pensant à la diversité linguistique et à la diversité du genre. Comme montre la Figure 1.1, il y avait les manuscrits, les incunables, et les imprimés. De plus, chaque type de document a porté plusieurs genres littéraires, y compris la poésie, le récit, et le traité. Dans le cas des imprimés, il y avait aussi des pièces de théâtre.

| | | | Ecriture | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Type | Genre | Siècle | Null | | antiqua | | cursive | | gothique | |
| manuscrit | Poésie | 13 | | | | | | | | 2 |
| | | 15 | | | | | | 3 | | |
| | Récit | 13 | | | | | | | | 7 |
| | | 14 | | | | | | 1 | | 6 |
| | | 15 | | | | | | 6 | | |
| | | 16 | | | | | | | | 1 |
| | Traité | 14 | | | | | | | | 1 |
| incunable | Poésie | 15 | | | | | | | | 2 |
| | Récit | 15 | | | | | | | | 5 |
| | | 16 | | | | 1 | | | | 1 |
| | Traité | 15 | | | | | | | | 2 |
| | | 16 | | | | 1 | | | | 2 |
| imprimé | Poésie | 16 | | | | 3 | | | | |
| | | 17 | | | | 7 | | | | |
| | | 18 | | | | 4 | | | | |
| | Récit | 16 | | 1 | | 2 | | 1 | | 2 |
| | | 17 | | 1 | | 9 | | | | |
| | | 18 | | | | 6 | | | | |
| | Théâtre | 16 | | | | 2 | | | | |
| | | 17 | | | | 5 | | | | |
| | | 18 | | | | 3 | | | | |
| | Traité | 16 | | | | 2 | | | | 1 |
| | | 17 | | | | 3 | | | | |
| | | 18 | | | | 12 | | | | |

FIGURE 1.1 – Diversité linguistique et générique

Pour terminer, un autre souhait quant à la diversité du corpus a porté sur le lieu

de publication ou d'apparition du document, comme montre la Figure 1.2. La plupart des documents choisis de la base de données Gallica étaient sortis de Paris. Une autre partie importante est venue de Lyon. Il y avait aussi un effort d'inclure les manuscrits, les incunables, et les imprimés écrits en français qui sont venus des villes hors de la France, tel que Londres, Amsterdam, et Rome.


FIGURE 1.2 – Diversité géographique

Après avoir établi le corpus, les pages ciblées dans chaque document étaient transcrites par des vacataires en utilisant l'interface graphique *eScriptorium*, qui permet à la fois la transcription du texte et la segmentation de la page. La segmentation de la page se faisant en mettant les étiquettes précises sur les masques des lignes et les bloques de texte ; ces étiquettes suivaient le vocabulaire *SegmOnto*. Ensuite, les vérités de terrain produites avec l'interface graphique *eScriptorium* étaient transférées vers les dépôts Github du bon siècle. L'outil `HTRVX` du projet HTR-United a analysé toute transcription transféré. L'outil s'est alerté aux erreurs potentielles de la segmentation et il a facilité le nettoyage des données.

## 1.3 Les prédécesseurs du projet

Comme expliqué avant, *Gallic(orpor)a* a rassemblé les recherches de plusieurs projets précédents. Il a profité des glossaires codicologiques, des progrès dans la prédiction et la segmentation des documents, des progrès dans l'analyse linguistique, et des catalogues

des données. Le glossaire *SegmOnto* se servait à harmoniser les vérités de terrain pour les manuscrits et les imprimés transcrits. Les modèles HTR étaient à la fois un support à la production des vérités de terrain, en faisant en premier temps une transcription préliminaire, et le but du projet, étant de produire les modèles entraînés sur le vocabulaire *SegmOnto*. Le catalogue HTR-United, spécifiquement son outil `HTRVX`, a surveillé l'harmonisation des transcriptions produites comme des vérités de terrain. [7] En outre, le catalogue HTR-United a publié gratuitement les vérités de terrain du projet *Gallic(orpor)a* pour que d'autres projets et d'autres modèles HTR puissent en profiter. [8] Pour terminer, l'analyse linguistique était aussi un objectif du projet *Gallic(orpor)a*, et la mise en œuvre de cet aspect comptait sur les modèles de la langue française construits par les chercheurs de l'équipe ALMAnaCH.

### Le glossaire codicologique

Le projet *Gallic(orpro)a* a harmonisé ses données selon le vocabulaire *SegmOnto*, qui est expliqué en détail dans le chapitre 2. Ayant géré les données produites selon cette codicologie, j'ai aussi contribué à l'élaboration et le perfectionnement du vocabulaire. La décision d'utiliser le syntaxe descriptif des lignes et des zones de *SegmOnto* a été prise bien en avance de la mise en œuvre du projet *Gallic(orpor)a*. La généralité du vocabulaire était déterminée d'être conforme à la diversité ciblée des documents traités dans le cadre du projet. Puisque *Gallic(orpor)a* visait à livrer un prototype d'un pipeline qui pourrait traiter tout document source numérisé, la généralisation des étiquettes appliquées aux lignes et aux zones était impérative, et le vocabulaire de *SegmOnto* était jugé la meilleure solution.

### La segmentation et la prédiction du texte

Les progrès de la reconnaissance du texte sont expliqués en détail dans le chapitre 3. Un logiciel HTR commence par la segmentation de la page, et dès qu'il sait où se trouvent les caractères, les mots, et les lignes du texte il le prédit à partir de l'écriture. Ces deux tâches se font selon les compétences qu'il a appris lors de son entraînement. Dans le but de produire les vérités de terrain pouvant entraîner les modèles HTR pour les manuscrits, les incunables, et les imprimés dans la base de données Gallica, le projet *Gallic(orpor)a* a profité de l'expertise de Simon Gabay et d'Ariane Pinche, qui s'occupaient de la relecture des vérités de terrain et la gestion des corpus d'or.

7. Thibault Clérice et Ariane Pinche. *HTRVX, HTR Validation with XSD*. Version 0.0.1. Sept. 2021. DOI : 10.5281/zenodo.5359963. URL : https://github.com/HTR-United/HTRVX (visité le 12/08/2022).

8. Alix Chagué et Thibault Clérice. « Sharing HTR Datasets with Standardized Metadata : The HTR-United Initiative ». In : Documents Anciens et Reconnaissance Automatique Des Écritures Manuscrites. 23 juin 2022. URL : https://hal.inria.fr/hal-03703989 (visité le 12/08/2022).

Les progrès dans la prédiction du texte sur les imprimés de l'Ancien Régime ainsi que son analyse ont aidé le projet *Gallic(orpor)a*. Par rapport aux progrès dans l'OCR des imprimés françaises de l'Ancien Régime, Gabay a entraîné les modèles sur les vérités de terrain des imprimés du XVIe au XVIIIe siècle.[9] En collaboration avec d'autres chercheurs, il a travaillé sur le jeu de données OCR17+ qui a fourni des vérités de terrain des imprimés du XVIIe siècle.[10] Pour tester le pipeline, dans l'attente des modèles nouvellement entraînés sur les données produites dans le cadre de *Gallic(orpor)a*, j'ai utilisé un modèle de segmentation et un modèle d'HTR que Gabay a développé dans le cadre de son projet *E-ditiones*.[11]

Pour la reconnaissance du texte sur les documents médiévaux, le projet CREMMA-Lab est clef. Géré dans le cadre des études postdoctorales d'Ariane Pinche, le Consortium Reconnaissance d'Écriture Manuscrite des Matériaux Anciens, ou CREMMA, est un dépôt des images et leurs transcriptions corrigées à la main, c'est-à-dire des vérités de terrain. Afin d'entraîner un modèle HTR, il faut un jeu des vérités de terrain.[12] Le projet CREMMALab fournit un jeu des vérités de terrain de 13 manuscrits médiévaux qui se composent de 21 656 lignes de texte transcrites.[13] Sur les données du projet, Pinche a entraîné un modèle HTR qui est désormais disponible sur Zenodo.[14] Le projet *Gallic(orpor)a* en a profité pour aider à la création des vérités de terrain pour les manuscrits médiévaux de son propre jeu de données.

### L'harmonisation et la partage des données

Le projet HTR-United mis en commun les vérités de terrain générées par tout projet *open source*.[15] Sa base de données, gratuitement mise en ligne par GitHub, contient les images et leurs transcriptions faites par plusieurs projets de recherche, et elle porte sur

9. Simon Gabay et al. « Standardizing Linguistic Data : Method and Tools for Annotating (Pre-Orthographic) French ». In : *Proceedings of the 2nd International Digital Tools & Uses Congress (DTUC '20)*. Hammamet, Tunisia, oct. 2020. doi : 10.1145/3423603.3423996. url : https://hal.archives-ouvertes.fr/hal-03018381 (visité le 12/08/2022).

10. Simon Gabay, Thibault Clérice et Christian Reul. *OCR17 : Ground Truth and Models for 17th c. French Prints (and Hopefully More)*. Mai 2020. url : https://hal.archives-ouvertes.fr/hal-02577236 (visité le 12/08/2022).

11. Simon Gabay. *E-Ditiones, 17th c. French Sources*. Nov. 2018. url : https://hal.archives-ouvertes.fr/hal-02388415 (visité le 10/08/2022).

12. Alix Chagué, Thibault Clérice et Laurent Romary. « HTR-United : Mutualisons La Vérité de Terrain ! » In : *DHNord2021 - Publier, Partager, Réutiliser Les Données de La Recherche : Les Data Papers et Leurs Enjeux*. Lille, France : MESHS, nov. 2021. url : https://hal.archives-ouvertes.fr/hal-03398740 (visité le 10/08/2022).

13. Ariane Pinche et Jean-Baptiste Camps. « CremmaLab Project : Transcription Guidelines and HTR Models for French Medieval Manuscripts ». In : *Colloque "Documents Anciens et Reconnaissance Automatique Des Écritures Manuscrites"*. Paris, France, juin 2022. url : https://hal.archives-ouvertes.fr/hal-03716526 (visité le 10/08/2022).

14. Ariane Pinche. « HTR model Cremma Medieval ». In : (21 juin 2022). doi : 10.5281/zenodo.6669508. url : https://zenodo.org/record/6669508 (visité le 10/08/2022).

15. Alix Chagué et al. *HTR-United/Htr-United : V0.1.28*. Zenodo, 10 août 2022. doi : 10.5281/zenodo.6979746. url : https://zenodo.org/record/6979746 (visité le 12/08/2022).

les documents de plusieurs périodes historiques et écritures. Un modèle HTR peut être entraîné sur ces jeux de données. Par exemple, Alix Chagué a entraîné un modèle HTR sur les vérités de terrain du *LECTAUREP Project*, soutenu par Inria et les Archives Nationales, qui sont mis en commun sur la base de données HTR-United. [16]

Dans l'esprit de la science ouverte, le projet *Gallic(orpor)a* a transféré toute vérité de terrain de ses dépôts GitHub vers le catalogue HTR-United. À la fin du stage, en juillet 2022, Ariane Pinche et Thibault Clérice ont entraîné un modèle pour les manuscrits médiévaux en utilisant les transcriptions que l'équipe du projet *Gallic(orpor)a* ont produites. Ces vérités de terrain sont désormais mises en commun sur HTR-United et Pinche et Clérice ont lié le premier modèle publié du projet *Gallic(orpor)a* avec le catalogue HTR-United et le dépot du projet CREMMA (Consortium Reconnaissance d'Écriture Manuscrite des Matériaux Anciens). [17] La partage des données du projet est l'un de ses objectifs.

Ainsi qu'à contribuer au catalogue, le projet *Gallic(orpor)a* a aussi profité des outils de HTR-United. Le dernier met en commun des outils qui ont pour but d'harmoniser les données ajoutés à son catalogue. Ces outils peuvent être intégrés dans un *workflow* de GitHub, ce que l'équipe de *Gallic(orpor)a* a fait. L'un de ces outils est `HTRVX`, qui se prononce comme le personnage Asterix, et il a rendu possible à l'équipe du projet nettoyer les transcriptions sorties de *eScriptorium*. [18] En exemple, `HTRVX` relit les transcriptions et les cherche pour les erreurs communes. L'existence d'un tel outil et sa disponibilité gratuite grâce au projet HTR-United a beaucoup aidé le projet *Gallic(orpor)a*.

### L'analyse linguistique

Après l'extraction et le nettoyage des données des documents source de Gallica, le projet *Gallic(orpor)a* a envisagé à analyser le texte. Dans cet objectif, il a profité des progrès dans l'analyse linguistique des anciens états de la langue française. L'analyse linguistique du français de l'Ancien Régime, tel que ce qui se voit dans les écrits de Molière, est un domaine de recherche actuellement en plein développement. Depuis une dizaine d'années, les chercheurs dans la linguistique computationnelle ont élaboré des outils pour analyser le français autre que le français contemporaine, dont les recherches sont déjà animées par l'application commerciale et les jeux de données plus nombreuses.

L'histoire de l'analyse linguistique et du Traitement automatique des langues (TAL) est hors de ce mémoire. Néanmoins, les projets qui ont précédés *Gallic(orpor)a* et sur lesquels il a compté méritent de discussion. Achim Stein a bordé le sujet de l'analyse linguistique du français du Moyen Âge dans son article de 2013. [19] Stein a montré que, pour les

16. Alix Chagué. *LECTAUREP Contemporary French Model (Administration)*. Zenodo, 12 mai 2022. URL : `https://zenodo.org/record/6542744` (visité le 12/08/2022).

17. Ariane Pinche et Thibault Clérice. *HTR-United/Cremma-Medieval : Cortado 2.0.0*. Zenodo, 11 juill. 2022. DOI : `10.5281/zenodo.6818057`. URL : `https://zenodo.org/record/6818057` (visité le 12/08/2022).

18. Clérice et Pinche, *HTRVX, HTR Validation with XSD*.

19. Achim Stein et Sophie Prévost. *Syntactic Annotation of Medieval Texts : The Syntactic Re-*

chercheurs qui ont débuté d'appliquer les progrès dans l'analyse linguistique à l'étude du français des anciens états, il faut faire attention aux propriétés syntaxiques et morphologiques propres à la langue. Du coup, une architecture qui pourrait parvenir aux résultats souhaités pour l'anglais du Moyen Âge n'aura pas forcement le même taux de réussite avec le français du Moyen Âge à cause de différences syntactiques et morphologiques dans la langue.

Achim Stein et Sophie Prévost ont créé un *treebank* pour l'ancien français, le *Syntactic Reference Corpus of Medieval French* (SRCMF), qui avance toujours l'analyse linguistique des anciens états du français. [20] En 2014, Prévost est des autres chercheurs, Gael Guibon, Isabelle Tellier, Matthieu Constant, et Kim Gerdes, ont ajouté aux conclusions de Stein que la variation lexique de l'ancien français pose aussi un défi à l'analyse linguistique. [21] En 2019, Mathilde Regnault, Prévost, et Éric Villemonte de La Clergerie ont utilisé le SRCMF avec le MCVF (Modéliser le changement : les voies du français) pour encore élaborer notre connaissance de l'évolution du français. [22] Telles études linguistiques ont tourné la terre pour que l'analyse du texte extrait des manuscrits médiévaux et des imprimés historiques puissent voir le jour aujourd'hui.

*Gallic(orpor)a* a profité des progrès dans l'analyse linguistique du français historique utilisé dans les imprimés et les manuscrits de ses corpus. Jean-Baptiste Camps, Simon Gabay, Paul Fièvre, Thibault Clérice, et Florian Cafiero ont créé *Deucalion* qui est un modèle TAL pour le français de l'Ancien Régime, entraîné sur les livrets des drames du XVIIe siècle. [23] Encore plus récent est le projet *FREEM*qui veut dire *French Early Modern* [24] Dans une présentation du projet à la conférence du Traitement Automatique des Langues Naturelles à Avignon en juin 2022, les auteurs ont décrit l'un des modèles qu'ils ont développé à partir de l'étude linguistique de l'ancien français.

---

*ference Corpus of Medieval French (SRCMF)*. Narr Verlag, 2013, p. 275. ISBN : 978-3-8233-6760-4. URL : `https://halshs.archives-ouvertes.fr/halshs-01122079` (visité le 10/08/2022).

20. Achim Stein et Sophie Prévost. *Syntactic Reference Corpus of Medieval French (SRCMF)*. Stuttgart : ILR University of Stuttgart, 2013. ISBN : 899-492-963-833-3. URL : `http://srcmf.org`.

21. Gaël Guibon et al. « Parsing Poorly Standardized Language Dependency on Old French ». In : *Thirteenth International Workshop on Treebanks and Linguistic Theories (TLT13)*. Sous la dir. de V. Henrich et al. Proceedings of the Thirteenth International Workshop on Treebanks and Linguistic Theories (TLT13). Tübingen, Germany, déc. 2014, p. 51-61. URL : `https://hal.archives-ouvertes.fr/hal-01250959` (visité le 10/08/2022).

22. Mathilde Regnault, Sophie Prévost et Éric Villemonte de La Clergerie. « Challenges of Language Change and Variation : Towards an Extended Treebank of Medieval French ». In : *TLT 2019 - 18th International Workshop on Treebanks and Linguistic Theories*. Paris, France, août 2019. URL : `https://hal.inria.fr/hal-02272560` (visité le 10/08/2022).

23. Jean-Baptiste Camps et al. « Corpus and Models for Lemmatisation and POS-tagging of Classical French Theatre ». In : *Journal of Data Mining & Digital Humanities* 2021 (Digital humanities in... 14 fév. 2021), p. 6485. ISSN : 2416-5999. DOI : `10.46298/jdmdh.6485`. arXiv : `2005.07505 [cs]`. URL : `http://arxiv.org/abs/2005.07505` (visité le 09/08/2022).

24. Simon Gabay et al. « Le Projet FREEM : Ressources, Outils et Enjeux Pour l'étude Du Français d'Ancien Régime ». In : *TALN 2022 - Traitement Automatique Des Langues Naturelles*. Sous la dir. d'Yannick Estève et al. Avignon, France : ATALA, juin 2022, p. 154-165. URL : `https://hal.archives-ouvertes.fr/hal-03701524` (visité le 10/08/2022).

> L'étude de la langue ne nécessitant pas uniquement un seul outil, mais toute une gamme de solutions, nous avons adopté une approche holistique du problème, en misant sur la création d'un modèle de langue dédié au français d'Ancien Régime, D'AlemBERT [25], qui devrait venir en soutien des différentes tâches de TAL envisagées. [26]

Le développement des nouveaux modèles TAL pour les anciens états du français a rendu possible la mise en œuvre d'un tel étape d'analyse au pipeline de *Gallic(orpro)a*.

## 1.4 Le pipeline

Ainsi que la création des vérités de terrain, qui est expliqué dans la section précédente de ce chapitre (Section 1.2), le projet *Gallic(orpor)a* avait pour but de créer un pipeline pouvant traiter tout document source de la base de données Gallica. En mettant en œuvre les modèles entraînés, il visait à générer une ressource lexicographique qui porte sur le document source. En commençant par les pages numérisés du document source, la ressource numérique présentera quatre types d'information :

1. les métadonnées à propos du document source
2. les données topologiques et linguistiques prédites par *l'Handwritten Text Recognition* (HTR)
3. le texte pré-éditorialisé, extrait du document
4. le texte analysé par les outils Traitement automatique des langues (TAL)

Chaque type d'information veut servir une utilité différente que la lectrice éventuelle ou le lecteur éventuel de la ressource pourrait désirer. Les métadonnées s'informent sur trois types de document concerné par le pipeline : (1) la ressource lexicographique qu'il créé, (2) le fac-similé numérique du document source, (3) le document source physique qui se conserve quelque part, sinon qui a été conservé avant sa disparition. Les données produites par les modèles HTR présentent la segmentation de la mise en page et la prédiction du texte. Ensuite, le texte pré-éditorialisé est présenté, sans sa segmentation, d'une manière plus convenable à l'analyse linguistique. Et en fin, la ressource présente son analyse du texte prédit grâce à l'application de certains modèles TAL.

Une visualisation du pipeline se voit dans le Figure 1.3. Elle montre en bleu les saisies des données, en vert les fichiers préliminaires que le pipeline construit, en orange sa première sortie, et en violet les divers moyens d'exploitation de sa sortie. Puisque le pipeline va générer des métadonnées et les données topologiques et linguistiques, il a

25. Simon Gabay et al. « From FreEM to D'AlemBERT ». In : *Proceedings of the 13th Language Resources and Evaluation Conference*. Marseille, France : European Language Resources Association, juin 2022. URL : `https://hal.inria.fr/hal-03596653` (visité le 10/08/2022).

26. Gabay et al., « Le Projet FREEM ».

besoin d'au moins deux saisies des données. L'un porte sur les métadonnées des trois types de document concernés (la ressource lexicographique elle-même, le fac-similé numérique du document source, le document source physique). L'autre saisie des données est l'image numérique elle-même. La première saisie peut se composer de plusieurs sources de données en ligne, afin de fournir à la ressource lexicographique autant de détail que possible. La deuxième saisie se compose du fac-similé numérique dans le format simple des images numériques, tel que JPEG, TIFF, ou PNG.

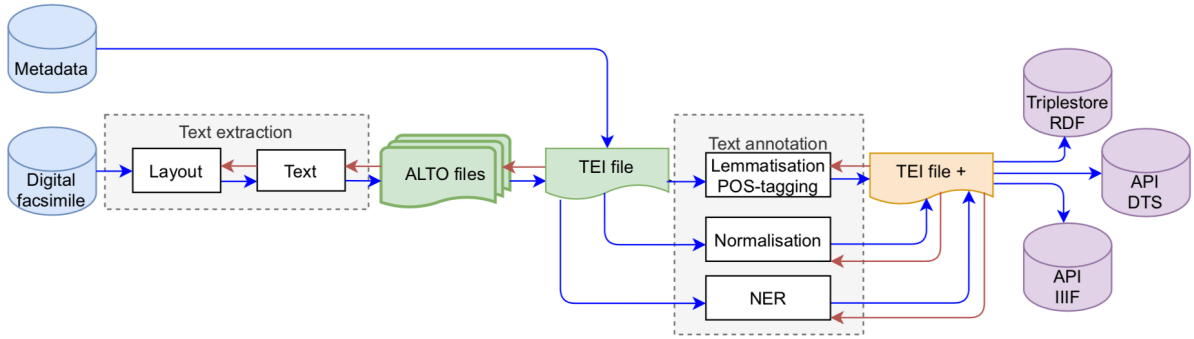



FIGURE 1.3 – Pipeline

En vert, la Figure 1.3 montre les premiers fichiers préliminaires créés par le pipeline, les fichiers ALTO. Ce format d'un fichier XML est un schéma particulièrement convenable à l'encodage de l'analyse de la mise en page que fait un modèle de segmentation et la prédiction du texte que fait un modèle HTR. Son acronyme veut dire en anglais *l'Analyzed Layout and Text Object*, ou la mise en page analysée et l'objet de texte. Comme se voit par la nature bipartite du nom *Analyzed Layout and Text Object*, les fichiers ALTO aligne la mise en page et le texte prédit. Le pipeline créé ce genre de fichier préliminaire en appliquant les modèles HTR aux données d'entrée visuelles.

Après la création des premiers fichiers préliminaires, le pipeline créé la première version du fichier TEI, qu'il va plus tard enrichir avec l'analyse linguistique du texte prédit. Ce premier fichier TEI occasionne la création des métadonnées. Comme montre la flèche bleue dans la Figure 1.3, les métadonnées sont récupérées depuis les sources en ligne et ensuite nettoyées et intégrées au fichier TEI. Le fichier TEI préliminaire récupère aussi les données des fichiers ALTO. Il est le combinaison de ces deux genres de données que rend le fichier TEI bien fait pour présenter toutes les données créés par le pipeline.

À la suite de la récupération, du nettoyage, et enfin de la transformation des données des deux sources pour les conformer au schéma TEI, le pipeline continue à traiter le texte qu'il a récupéré des fichiers ALTO. Le pipeline trait deux fois le texte prédit. En premier temps, il parse les données récupérées des fichiers ALTO et extrait toute ligne de texte imbriquée dans une zone qui fait partie du texte principal. Cela veut dire qu'il n'extrait pas de numéro de page, d'en-tête, etc. Les lignes de texte ainsi sélectionnées sont présentées comme le texte pré-éditorialisé du document source. Elles représentent une espèce de

transcription du texte, en ignorant la mise en page et les autres types d'écriture sur la page qui sont communiqués autre part dans le fichier TEI.

Ce texte pré-éditorialisé peut servir à l'analyse d'une utilisatrice ou d'un utilisateur qui veut appuyer sur le texte tel qu'il était dans le document source. Ce texte sert aussi à la génération d'un texte analysé par les outils TAL. Le résultat de ce dernier traitement est visualisé dans la Figure 1.3 comme la sortie en orange, le fichier TEI enrichi, *TEI file +*. Pour terminer, les données de la sortie peuvent être exploitées dans plusieurs formats, que la Figure 1.3 visualise en violet.

# Chapitre 2

# Au commencement, il y avait les *guidelines SegmOnto*

## 2.1 La problématique

Un manuscrit se définit par ses moyens de création. Étant rédigé à la main, souvent avant la croissance de l'imprimerie, un manuscrit est un objet singulier dans le monde. Il est vrai que d'autres ressources peuvent présenter le même texte, les mêmes images, ou bien la même musique que présent un manuscrit. Cependant, un manuscrit n'a pas d'autre exemplaire. Ses contenus sont réalisés par et se répandent dans sa constitution matérielle particulière. Un manuscrit est unique avec son écriture, parfois de plusieurs mains, ses fautes d'écriture, ses parties abîmées, décorées, ou révisées, sa provenance et son histoire en tant qu'objet rare qui se transfère entre des individus, des familles, et des organisations. De plus, chaque propriétaire peut encore modifier la ressource en ajoutant ou retirant des pages, en corrigeant ou blâmant du texte ou des images, ainsi qu'en changeant la reliure et les informations portant sur l'édition.

Pour étudier un manuscrit, il faut donc développer un vocabulaire qui sait décrire les divers aspects de l'objet composé. Après tout, le pouvoir de bien définir les termes d'une étude est à la base de l'analyse. Sans un vocabulaire bien élaboré et cohérent, les arguments d'une chercheuse ou d'un chercheur ne seront pas compréhensibles. L'harmonisation d'un vocabulaire est encore plus importante eu égard à la communication des découverts et à la collaboration entre plusieurs personnes, surtout si elles ont des spécialités différentes. Ces deux activités, la communication et la collaboration, sont fondamentales à la recherche et exigent donc l'élaboration d'un vocabulaire cohérent pour décrire des manuscrits.

Voici les défis de nommage dans l'étude des manuscrits et voici l'un des obstacles que le projet *Gallic(orpor)a* a essayé de franchir. Imaginons, par exemple, qu'on veut décrire le texte principal qui se trouve sur la page d'un document. On peut y appliquer l'étiquette descriptive *Main Zone* ou bien *Principal Text*. En lisant des articles scientifiques où cha-

cun utilise une étiquette différente, un humain arriverait à reconnaître que les étiquettes différentes parlent de la même partie de la page. Mais pour un outil informatique, si la même région d'une page ne porte pas la même étiquette, il n'arriverait pas à les associer sans être instruit à chercher plusieurs versions du même concept. Dans ce cas, l'analyse serait trop compliquée à effectuer à l'échelle. C'est ainsi que la description des manuscrits est importante au projet *Gallic(orpor)a*. Le projet *Gallic(orpor)a* cherchait donc à profiter d'un vocabulaire bien élaboré et cohérent, qui pourrait décrire soit les manuscrits, soit les imprimés historiques. Il y avait plusieurs projets qui avaient cherché à répondre à cette problématique, mais celui que le projet *Gallic(orpor)a* a choisi est le vocabulaire du projet *SegmOnto*.

## 2.2 Les solutions proposées

Plusieurs projets avaient proposé des solutions quant à la description normalisée des manuscrits. Hors de la France, les vocabulaires ont été élaborés notamment en anglais et pour les manuscrits médiévaux. Un exemple important est la base de données *DigiPal* (Digital Resource and Database of Palaeography, Manuscripts and Diplomatic), qui n'est plus mis à jour mais qui a été développé au sein du département des humanités numériques à King's College London. [1] L'un de ses auteurs, Peter Stokes, travaille actuellement en France et continue dans la même veine en contribuant au projet *SegmOnto* qui était développé dans un environnement francophone, même si son vocabulaire est rédigé en anglais. [2] Mais le projet *SegmOnto* n'est pas le premier projet français qui a essayé d'élaborer un lexique pour les documents historiques. Ni est le projet *DigiPal* le premier en Europe. Avant leur création, la codicologie en France et en Europe suivait le modèle de Denis Muzerelle et son *Vocabulaire codicologique*.

### 2.2.1 Le Vocabulaire international de la codicologie

En 1985, Denis Muzerelle a conçu un vocabulaire codicologique qui avait pour but de fournir des médiévistes avec des termes uniformes pouvant décrire les aspects d'un manuscrit. [3] Depuis l'apparition de son vocabulaire en français, des autres chercheurs sont venus pour adapter les termes de Muzerelle en d'autres langues. Marilena Maniaci

1. *DigiPal : Digital Resource and Database of Palaeography, Manuscripts and Diplomatic.* London. URL : `http://www.digipal.eu/`.

2. Simon Gabay, Jean-Baptiste Camps et Ariane Pinche. « SegmOnto ». In : *Création de Modèle(s) HTR Pour Les Documents Médiévaux En Ancien Français et Moyen Français Entre Le Xe-XIVe Siècle*. Paris, France : Ecole nationale des chartes | PSL, nov. 2021. URL : `https://hal.archives-ouvertes.fr/hal-03481089` (visité le 25/07/2022).

3. Dennis Muzerelle. *Vocabulaire Codicologique : Répertoire Méthodique Des Termes Français Relatifs Aux Manuscrits*. Rubricae 1. Paris : Éd. Cemi, 1985.

a publié une version du *Vocabulaire codicologique* pour l'italien en 1996. [4] Pilar Ostos, Luisa Pardo, et Elena Rodríguez en ont créé un pour l'espagnol l'année suivante. [5] Parfois appelé le *Vocabulaire international de la codicologie*, l'édition multilingue du *Vocabulaire codicologique* que Muzerelle a commencé en 1985 était maintenue jusqu'à l'édition d'une version 1.1. en 2002-2003. (besoin de citation)

### 2.2.2 La Codicologia

Aujourd'hui, la paléographie et l'étude des manuscrits peuvent profiter de l'application web *Codicologia* qui réunit le *Vocabulaire codicologique* ainsi que deux autres bases de données similaires : le projet multilingue *Lexicon* et le *Glossaire codicologique arabe*. Ses trois bases de données spécialisent dans divers écritures. Le *Vocabulaire codicologique* a été développé pour les manuscrits de l'écriture latin. Piloté par Philippe Bobichon, le projet *Lexicon* présente un vocabulaire en français pour décrire les manuscrits écrits en latin, roman, grec, hébreu, et arabe. [6] Un vocabulaire spécialisé plus profondément pour l'arabe a été élaboré dans le *Glossaire codicologique arabe* d'Anne-Marie Eddé et Marc Geoffroy. [7] Ce dernier a été conçu au sein de l'Institut de recherche et d'histoire des textes après les modèles de Muzerelle et le vocabulaire codicologique en arabe d'Adam Gacek. [8]

L'application web *Codicologia* rassemblent ces projets et présente un vocabulaire bien étendu. Par exemple, *Codicologia* fournit 15 termes pour décrire une faute d'écriture dans un manuscrit. Certains de ses termes possèdent eux-même plusieurs définitions que les divers bases de données fournissent. Le terme *caviarder*, par exemple, a une définition courte dans le vocabulaire français de Muzerelle.

> Supprimer un mot, un passage..., en le recouvrant largement d'encre, de façon à ce qu'il ne puisse être lu. [9]

Selon le *Lexicon* de Bibichon, par contre, le *caviarder* se définit d'une manière plus détaillé et vise à expliquer l'étymologie du mot afin de préciser son usage dans le cadre des manuscrits des divers écritures.

---

4. Marilena Maniaci. *Terminologia Des Libro Manoscritto*. Addenda 3. Rome : Istituto centrale per la patologia del libro, 1996.

5. Pilar Ostos, M. Luisa Pardo et Elena E. Rodríguez. *Vocabulario de codicologÍa : Versión Española Revisada y Aumentada Del Vocabulaire Codicologique*. Instrumenta Bibliologica. Madrid : Arco/Libros, 1997.

6. Philippe Bobichon. « Le Lexicon : Mise En Page et Mise En Texte Des Manuscrits Hébreux, Grecs, Latins, Romans et Arabes ». In : (2009), p. 81. URL : https://cel.archives-ouvertes.fr/cel-00377671 (visité le 25/07/2022).

7. « Glossaire codicologique français-arabe ». In : *Gazette du livre médiéval* 40.1 (2002), p. 79-80. URL : https://www.persee.fr/doc/galim_0753-5015_2002_num_40_1_1563 (visité le 25/07/2022).

8. Adam Gacek. *The Arabic Manuscript Tradition : A Glossary of Technical Terms and Bibliography*. Handbook of Oriental Studies 1, The Near and Middle East. Leiden : Brill, 2001. 269 p. ISBN : ISBN 90-04-12061-0.

9. Denis Muzerelle. *Caviarder*. In : *Codicologia*. Institut de recherche et d'histoire des textes, 2011. URL : http://codicologia.irht.cnrs.fr/theme/liste_theme/413#tr-868 (visité le 25/07/2022).

> Le mot [*caviarder*] apparaît en 1907 (noircir à l'encre) : il désigne alors un procédé appliqué par la censure russe, sous Nicolas Ier. Dans certains manuscrits grecs, le détail rempli d'encre est surmonté d'un point et d'un trait court destinés à le neutraliser. Ce procédé est très souvent utilisé parmi d'autres, pour la censure des manuscrits hébreux effectué sous l'autorité de l'inquisition, en Italie, à la fin du xvie siècle et au début du xviie. [10]

Étant élaboré à partir d'un corpus très diversifié, le *Lexicon* de Bibichon a moins de termes qu'a le *Vocabulaire codicologique* mais ses termes sont plus généralisés. Le vocabulaire de Muzerelle, par contre, fait plus de distinctions entre les aspects d'un manuscrit et donc a plus de termes distincts par rapport aux deux autres vocabulaires de l'application *Codicologia*.

En réunissant les trois bases de données, sans privilégier aucun, *Codicologia* présente un vocabulaire codicologique vraiment vaste. Cependant, l'application *Codicologia*, comme toutes ses bases de données, vise à répondre au manque de cohérence dans la manière par laquelle la communauté scientifique décrit les manuscrits. Le grandeur de son vocabulaire pose un problème à cet objectif. Ayant plus de deux milles termes en français—certains d'entre eux ont eux-même plusieurs définitions—la solution proposée par *Codicologia* livre un vocabulaire bien harmonisé et documenté mais trop étendu pour être appliqué à l'échelle dans une approche informatique.

Sans un corpus d'entraînement gigantesque, qui couterait une somme énorme, l'apprentissage automatique ne peut pas faire de distinction au niveau des termes conçus par Muzerelle et les autres auteurs des bases de données de *Codicologia*. Aujourd'hui, un modèle ne peut pas s'entraîner sur des milles des étiquettes possibles et arriver à distinguer entre, par exemple, 15 types de faute d'écriture. Un humaine peut le faire, et pour cette raison les bases de données de *Codicologia* sont utiles. Mais leurs vocabulaires ne conviennent pas bien à une approche informatique.

## 2.3 Les *guidelines* de *SegmOnto*

Le projet *SegmOnto* propose un vocabulaire plus petit qui peut pourtant décrire une grande diversité de documents historiques, y compris les manuscrits et les imprimés. Cet objectif est encore plus compliqué à achever qu'un vocabulaire spécialisé aux manuscrits. Décrire les documents d'une diachronie longue, et sans préférence d'une écriture en particulier, exige un équilibre délicat entre la généralité et la particularité. Pour y arriver, les *guidelines* du projet *SegmOnto* limite le nombre de termes dans son vocabulaire, sans en priver aucun d'une identité distincte.

10. Philippe Bobichon. *Caviarder*. In : *Codicologia*. Institut de recherche et d'histoire des textes, 2011. URL : `http://codicologia.irht.cnrs.fr/theme/liste_theme/413#tr-868` (visité le 25/07/2022).

Les *guidelines* se divisent en deux catégories : les «  zones » et les «  lines ». La première parle des régions sur la page, y compris les régions de texte et les régions sans texte, tel qu'une image. Pour la plupart de temps, la catégorie de la ligne veut décrire les différents types de lignes de texte. Mais une ligne du vocabulaire *SegmOnto* peut aussi tracer la ligne d'une partition musicale ou une ligne réelle sur la page qui n'oriente pas des autres systèmes d'écriture, telle qu'une ligne qui divise la page en deux. Chacune de ces deux catégories se compose d'une liste des étiquettes, et chacune d'elles cherche à parvenir à l'équilibre entre la généralité et la particularité. Une étiquette devrait pouvoir être appliquée à soit un manuscrit, soit un imprimé, de peu importe quelle langue et quelle écriture.

### 2.3.1 Les zones

— **CustomZone** : une zone qui ne convient pas à aucune d'autres catégories de zone.
— **DamageZone** : une zone qui contient des marques de dégâts sur le document source, tel qu'un trou.
— **DecorationZone** : une zone qui contient un élément graphique, y compris de la peinture et les petits dessins dans la marge de la page.
— **DigitizationArtefactZone** : une zone qui contient un item qui n'appartient pas au document source mais est lié au processus de la numérisation, tel qu'une règle pour montrer la mesure du document.
— **DropCapitalZone** : une zone qui contient une initiale ; l'initiale peut prendre l'espace de plusieurs lignes de texte ou porter une décoration importante, tel que de l'historicisation, l'ornementation, ou des dessins.
— **MainZone** : une zone qui contient le texte principal du document source.
— **MarginTextZone** : une zone qui contient le texte dans la marge du document source.
— **MusicZone** : une zone qui contient une partition musicale.
— **NumberingZone** : une zone qui contient des numéros de page, y compris les numéros rédigés en chiffres romans.
— **QuireMarksZone** : une zone qui contient des notes en bas page destinées à la fabrication du document source pour garder les pages dans le bon ordre.
— **RunningTitleZone** : une zone qui contient une version du titre du document ou d'une section du document qui se trouve en tête de la page.
— **SealZone** : une zone qui contient un sceau sur le document source.
— **StampZone** : une zone qui contient l'empreint d'un tampon sur le document source.
— **TableZone** : une zone qui contient une table.

— **TitlePageZone** : une zone souvent sur l'une des premières pages du document source qui contient toutes les informations concernant le titre et l'édition du document.

### 2.3.2 Les lignes

— **CustomLine** : une ligne qui ne convient pas à aucune d'autres catégories de ligne.
— **DefaultLine** : une ligne qui contient du texte attendu dans la zone.
— **DropCapitalLine** : une ligne qui contient l'initiale.
— **HeadingLine** : une ligne qui contient le texte d'un titre, tel que celui d'une section ou d'un chapitre.
— **InterlinearLine** : une ligne qui traverse la page pour marquer une limite.
— **MusicLine** : une ligne de la portée d'une partition.

# Chapitre 3

# Qu'est-ce que l'HTR ?

Qu'est-ce qu'est l'*Handwritten Text Recognition* ou la reconnaissance automatique d'écriture manuscrite ? L'*Handwritten Text Recognition* (HTR) est l'approche que l'équipe de *Gallic(orpor)a* a choisi pour prédire le texte écrit sur les fac-similés numériques de Gallica, y compris les manuscrits et les imprimés. À partir des pages numérisées, un logiciel HTR peut décomposer l'image en segments et en lignes de texte ainsi que reconnaître l'écriture dedans. Dit simplement, l'HTR parvenir à réaliser un texte bien structuré selon la logique de la page à partir des images numériques du texte. Cette technologie était utile au projet *Gallic(orpor)a* dont le pipeline avait pour but de saisir les images numériques et d'arriver à une transcription ainsi qu'aux versions du texte pré-éditorialisés et annotées.

## 3.1 Les origines de l'HTR

L'*Handwritten Text Recognition* a évolué à partir des recherches sur l'*Optical Character Recognition.*[1] Depuis ses origines, cette technologie précédente, l'OCR, avait pour but la reconnaissance du texte imprimé, y compris la police du texte. N'étant pas développé pour l'écriture manuscrite, l'OCR ne convient pas assez bien à la prédiction du texte sur les documents manuscrits. Par conséquent, la technologie de l'HTR a été développé, afin de surmonter les polices qui imposent des limites à l'OCR. La première conférence Frontiers in Handwriting Recognition a eu lieu en 1990 et aujourd'hui, le champs de recherche de l'HTR est en plein développement. Bien qu'il se soit développé en parallèle avec l'OCR depuis quelques décennies, le dernier l'a précédé et a préparé le terrain pour les nouveautés dans l'HTR à la fin du siècle précédent.

Les origines de l'OCR remontent au dix-neuvième siècle. Les progrès les plus importants de cette technologie ont eu lieu il y a plus d'un demi-siècle aux États-Unis. En

1. Sebastiano Impedovo. « More than Twenty Years of Advancements on Frontiers in Handwriting Recognition ». In : *Pattern Recognition*. Handwriting Recognition and Other PR Applications 47.3 (1er mars 2014), p. 916-928. issn : 0031-3203. doi : 10.1016/j.patcog.2013.05.027. url : https://www.sciencedirect.com/science/article/pii/S0031320313002513 (visité le 29/08/2022).

effet, le développement des logiciels OCR a été réalisé dans le cadre de l'amélioration du traitement en masse des données numérisées. Dans les années soixante, le besoin de gérer et traiter de grandes quantités de données est devenu de plus en plus pressant notamment dans les grandes entreprises, dont les banques. [2] La numérisation des journaux et le traitement en masse des lettres à la poste furent aussi l'un des contextes importants du développement de la reconnaissance automatique du texte sur des images.

Dans les années 1960, de plus en plus de sociétés souhaitaient adopter les méthodes de traitement des données assisté par ordinateur. Ce traitement a exigé l'encodage des données dans un format directement exploitable par ordinateur. Il y avait alors deux moyens pour la saisie des données : (i) la carte perforée dans laquelle une machine faisait des trous qu'un ordinateur savait lire, et (ii) la bande magnétique sur laquelle une machine imprimait des bytes, la plus petite unité exploitable par ordinateur. Dans les deux cas, le matériel a changé, du carton à la bande magnétique, mais le mécanisme binaire d'analyse restait : plein ou vide pour le carton, positif ou négatif pour la bande. Ni l'un ni l'autre technologie a permis de traiter les données qu'ils ont enregistrés.

L'*Optical Character Recognition* s'appuyait sur la bande magnétique mais il a ajouté la possibilité de traiter les données enregistrées. Le scanner saisit un objet, tel qu'une page, et renvoie une image numérique qui sert à la saisie du logiciel OCR. Cette acquisition des données automatique est différente que l'acquisition manuelle des méthodes d'encodage précédentes, qui comptaient sur un individu (souvent une femme) à saisir des données en tapant le texte à la perforatrice ou par le clavier du *Magnetic Tape/Selectric Typewriter* (MT/ST). [3] L'automatisation d'acquisition des données, grâce au scanner, est un point important dans l'évolution de la reconnaissance du texte.

En 1971, le chercheur Ben R. Schneider a comparé l'OCR et les deux autres moyens courants de l'encodage du texte dans les années 1960 : le MT/ST et la carte perforée. [4] Schneider a déterminé que l'OCR n'avait pas encore surpassé l'appareil MT/ST d'IBM parce que, malgré sa saisie automatique des données, il exigeait toujours beaucoup de correction à la main, car contrairement aux deux autres méthodes, les données d'entrée du traitement OCR n'étaient pas créées par un humain mais par un scanner. L'appareil MT/ST avait donc une exactitude supérieure à l'OCR à l'époque.

En outre, un logiciel OCR était limité par son jeu de données de polices, puisqu'il prédit du texte en faisant un comparaison entre un caractère qu'il a mémorisé et le motif

2. Deepa Berchmans et S S Kumar. « Optical Character Recognition : An Overview and an Insight ». In : *2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT)*. 2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT). Juill. 2014, p. 1361-1365. DOI : 10.1109/ICCICCT.2014.6993174.

3. Thomas J. Misa. *Gender Codes : Why Women Are Leaving Computing*. John Wiley & Sons, 14 sept. 2011. 326 p. ISBN : 978-1-118-03513-9. Google Books : EjDYh_KHls8C, p. 85-87.

4. Ben R. Schneider. « The Production of Machine-Readable Text : Some of the Variables ». In : *Computers and the Humanities* 6.1 (sept. 1971), p. 39-47. ISSN : 0010-4817, 1572-8412. DOI : 10.1007/BF02402324. URL : http://link.springer.com/10.1007/BF02402324 (visité le 02/08/2022).

qu'il a reconnu sur l'image. Même aujourd'hui l'OCR se distingue du champs parallèle de l'HTR par le fait qu'il compte sur une base de données des polices. Contraire à l'OCR dans les années 1960, l'appareil MT/ST pouvait encoder des documents des diverses polices en comptant sur le discernement d'un être humain lors de la saisie des données.

En 1977, Gian Piero Zarri, en résumant l'état de l'art de l'OCR, a remarqué que la reconnaissance du texte sur les manuscrits n'était pas encore faisable.

> Rappelons que la reconnaissance optique des caractères permet la lecture directe du texte par un « scanner » qui se charge d'effectuer le transfert sur bande magnétique ; le texte doit être composé avec des caractères de type « imprimerie » ou « machine à écrire », car la lecture de caractères « manuels » ne semble pas encore actuellement complètement sortie de la phase expérimentale. [5]

Dans les mêmes années, le chercheur américain Ray Kurzweil fait le même constat : *Kurzweil* ou le *Kurzweil Reading Machine* (KRM). [6] L'OCR peut reconnaître plusieurs polices. Cependant, comme l'indique Zarri, l'OCR reste sous la dépendance de la reconnaissance des polices et donc ne peut pas encore parvenir à la prédiction de l'écriture manuscrite.

## 3.2 Le fonctionnement général de l'HTR

### 3.2.1 L'image numérique

L'un des objectifs de l'HTR est d'imiter l'œil humain et de reconnaître les lignes et les points d'une écriture sur une image numérisée contenant du texte. Une image numérique consiste en un canevas (*canvas*) qui se compose de petits carrés apellés des pixels. Issu de l'anglais, le mot pixel signifie *picture element* et désigne l'élément minimal d'une image numérique. Les pixels sont stockés sous le format *bitmap* ou BMP et chaque pixel peut compter 1 bit, 4 bits, 8 bits, ou 24 bits selon l'encodage de l'image. Vus ensemble, un groupe de pixels peut donner l'impression des courbes d'un objet ou d'un caractère. L'exemple de figure 3.1 montre la diagonale de la lettre « A » écrite en crayon rouge.

Un système informatique stock une image numérique dans un format qui la décompose en unités de pixel. Chaque pixel se compose de la superposition de l'intensité de trois couches de couleur : rouge, vert, et bleu. (cf. Figure 3.2). Le premier entier qui se trouve dans la donnée tripartite d'un pixel défini l'intensité de la couleur rouge ; le deuxième

5. Gian Piero Zarri. « Quelques aspects techniques de l'exploitation informatique des documents textuels : saisie des données et problèmes de sortie ». In : *Publications de l'École Française de Rome* 31.1 (1977), p. 399-413. URL : `https://www.persee.fr/doc/efr_0000-0000_1977_act_31_1_2286` (visité le 01/08/2022).

6. Gregory Goodrich. « Kurzweil Reading Machine : A Partial Evaluation of Its Optical Character Recognition Error Rate ». In : *Journal of Visual Impairment and Blindness* (12 jan. 1979).

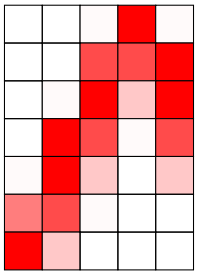


FIGURE 3.1 – Une couleur par pixel

entier celle de la couleur vert et le troisième celle de la couleur bleu. Le système s'appelle donc « RVB » puisqu'en effet, chaque pixel est une liste des trois couches des couleurs rouge, vert, et bleu. Avec un peu de distance, l'œil humain distingue les formes qui composent une image à partir de ce canevas. L'HTR a besoin des algorithmes pour identifier et reconnaître les pixels contigus formant un caractère ou un mot, ce que l'œil fait avec facilité.

| | | | | |
|---|---|---|---|---|
| 255,255,255 | 255,255,255 | 255,250,250 | 255,0,0 | 255,250,250 |
| 255,255,255 | 255,255,255 | 255,75,75 | 255,75,75 | 255,0,0 |
| 255,255,255 | 255,250,250 | 255,0,0 | 255,200,200 | 255,0,0 |
| 255,255,255 | 255,0,0 | 255,75,75 | 255,250,250 | 255,75,75 |
| 255,250,250 | 255,0,0 | 255,200,200 | 255,255,255 | 255,200,200 |
| 255,125,125 | 255,75,75 | 255,250,250 | 255,255,255 | 255,255,255 |
| 255,0,0 | 255,200,200 | 255,255,255 | 255,255,255 | 255,255,255 |

FIGURE 3.2 – Donnée RVB

### 3.2.2 Le *preprocessing*

Avant de faire des prédictions sur une image de texte, tout logiciel HTR ou OCR a besoin des étapes préliminaires qui sont connus ensemble sous le nom anglais *preprocessing*. L'une des étapes est souvent le redressement de l'image dans laquelle l'image s'ajuste jusqu'a les lignes de texte sont perpendiculaires. Tels traitements incluent le *dewarping*, que le projet *AGODA* a utilisé afin de corriger les courbes sur les pages numérisées des journaux pour que les lignes de texte soient toutes horizontales. [7]

Avant le redressement de l'image, une étape qui se fait souvent est la binarisation. Depuis quelques années, cette étape n'est plus nécessaire pour l'HTR dont les logiciels réussissent à redresser et segmenter l'image en traitant directement les pixels RVB. Cependant, la binarisation est toujours courante pour les logiciels OCR contemporains. [8] Comme expliquent Patrick Jentsch et Stephan Porada, « *The idea is to only extract the pixels which actually belong to the characters and discard any other pixel information which, for example, is part of the background.* [9] » La binarisation trie les pixels d'une image en deux classes : l'arrière-plan (*background* en anglais) et le premier plan (*foreground* en anglais).

#### Le niveau de gris

Normalement pour binariser une image, on veut remplacer la valeur composée d'un pixel, c'est-à-dire les trois degrés du rouge, du vert, et du bleu, avec la valeur simple d'un décimal. Ce décimal veut représenter l'échelle des gris dans le pixel. En anglais, ce traitement s'appelle le *grayscale* ou le niveau de gris en français. Aujourd'hui les langages de programmation ont souvent des librairies qui fournissent des méthodes pour rendre une image en niveau de gris automatiquement. Mais dans la figure 3.3, nous donnons un calcul simple qui permet le traitement des niveaux de gris. On commence toujours avec la donnée tripartite du pixel, qu'on veut remplacer par une seule valeur. Représentons chaque partie de la donnée du pixel par les variables $p$ :

$$p_1, p_2, p_3$$

Dans un premier temps, on prend la moyenne des degrés de la couleur, c'est-à-dire la moyenne de $p$ ou $\bar{p}$. La Figure 3.5a montre le résultat de ce calcul superposé à l'image

7. Aurélien Pellet et Marie Puren. « Le Projet AGODA. Océrisation Des Débats Parlementaires Français de La Troisième République : Problèmes, Défis et Perspectives ». In : Séminaire OMNSH-Epitech : Le Numérique Au Service Des Sciences Humaines et Sociales. Le Kremlin-Bicêtre, France, avr. 2022. URL : `https://hal.archives-ouvertes.fr/hal-03651146` (visité le 29/08/2022).

8. Patrick Jentsch et Stephan Porada. « From Text to Data Digitization, Text Analysis and Corpus Linguistics ». In : *Digital Methods in the Humanities : Challenges, Ideas, Perspectives*. Sous la dir. de Silke Schwandt. Bielefeld University Press, 2021.

9. Ibid., p. 107.

originale.

$$\bar{p} = \sum_{i=1}^{3} \frac{1}{3} p_i = \frac{1}{3}(p_1 + p_2 + p_3)$$

Ensuite, on récupère $\bar{p}$ et la divise par la valeur maximale du $p$, qui est 255 parce que le degré du rouge, vert, ou bleu d'un pixel ne monte qu'à 255.

| Donnée tripartite du pixel $p_1, p_2, p_3$ | Moyenne du pixel $\sum_{i=1}^{3} p_i$ | Valeur niveau de gris du pixel $\frac{\sum_{i=1}^{3} p_i}{\max\{p_i\}}$ |
|---|---|---|
| 255,000,000 | $\bar{p} = \frac{255+0+0}{3} = 85$ | $\frac{\bar{p}}{255} = 0.33$ |
| 255,075,075 | $\bar{p} = \frac{255+75+75}{3} = 135$ | $\frac{\bar{p}}{255} = 0.53$ |
| 255,125,125 | $\bar{p} = \frac{255+125+125}{3} = 168.33$ | $\frac{\bar{p}}{255} = 0.66$ |
| 255,200,200 | $\bar{p} = \frac{255+200+200}{3} = 218.33$ | $\frac{\bar{p}}{255} = 0.86$ |
| 255,250,250 | $\bar{p} = \frac{255+220+220}{3} = 251.67$ | $\frac{\bar{p}}{255} = 0.99$ |
| 255,255,255 | $\bar{p} = \frac{255+255+255}{3} = 255$ | $\frac{\bar{p}}{255} = 1.00$ |

FIGURE 3.3 – Évaluation des pixels

### Le seuil

Il y a plusieurs techniques de binarisation mais toute a besoin d'un seuil (*threshold* en anglais). Le seuil nous permet à trier les pixels en les deux classes : le *background* et le *foreground*. Nos yeux font cette étape facilement, mais un ordinateur a besoin d'un algorithme. L'un des algorithmes les plus courant pour définir le seuil a été élaboré en 1979 par le chercheur Nobuyuki Otsu. [10]

La méthode Otsu de seuillage reste toujours courante dans l'HTR [11] et elle fait partie de la librairie Python `numpy`.. [12] Elle examine la variance entre les deux classes (*background* et *foreground*) pour déterminer un seuil idéal pour le jeu de données. Visualisées dans un histogramme, comme on voit dans la figure 3.4, les données d'un jeu de pixels devraient se lever dans deux sommets et, idéalement, une baisse profonde devrait les diviser. Cette variance veut dire qu'il y a dans l'image une distinction importante entre les contours d'un caractère et l'arrière-plan de l'image. La méthode de seuillage examine la variance entre ces deux classes, le contour et l'arrière-plan, pour générer un seuil adapté aux données.

10. Nobuyuki OTSU. « A Threshold Selection Method from Gray-Level Histograms ». In : *IEEE Transactions on Systems, Man, and Cybernetics* 9.1 (jan. 1979), p. 62-66.

11. Imran-Ahmed SIDDIQI, Florence CLOPPET et Nicole VINCENT. « Writing Property Descriptors : A Proposal for Typological Groupings ». In : *Gazette du livre médiéval* 56.1 (2011), p. 42-57. DOI : 10.3406/galim.2011.1981. URL : https://www.persee.fr/doc/galim_0753-5015_2011_num_56_1_1981 (visité le 02/08/2022).

12. *Numpy : NumPy Is the Fundamental Package for Array Computing with Python.* Version 1.23.1.

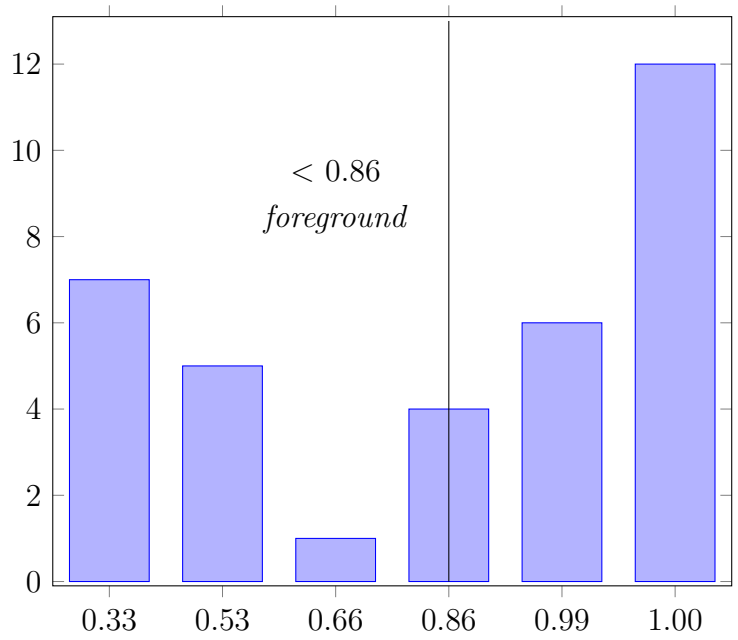



FIGURE 3.4 – Histogramme des valeurs niveau de gris des pixels

Le seuil sert à transformer la valeur numérique de chaque pixel soit en 0, soit en 1. Les pixels dont la valeur tombe au-dessous du seuil prennent la valeur 0 ; les autres, étant déterminés de faire partie du contour du caractère, prennent la valeur 1. La Figure 3.5c montre un exemple du résultat de ce triage. Ainsi, les logiciels OCR et HTR peuvent analyser les valeurs identiques et contiguës dans les données de l'image. Mais le logiciel n'est pas encore prêt à percevoir l'occurence d'un caractère.

### 3.2.3 Les tâches d'un logiciel HTR

Pour aboutir à une prédiction, le système d'HTR passe par trois étapes : la reconnaissance des différentes zones de la page, la reconnaissance des lignes où se trouve du texte, et la transcription du texte. L'analyse de la mise en page n'est pas obligatoire, mais les deux autres tâches sont essentielles. Chacune exige son propre modèle entraîné spécifiquement pour sa tâche. (cf. Figure 3.6)

#### La segmentation des zones de la page

La première tâche de la reconnaissance du texte est de localiser l'emplacement du texte. Cette étape s'appelle souvent la segmentation et elle est indispensable. [13] Selon son entraînement, un modèle de segmentation recherche les espaces entre les contours d'un caractère ou entre les lignes de texte. Si le modèle était entraîné pour reconnaître du texte écrit en japonais, par exemple, il rechercherait l'ensemble de caractères bordés à gauche et

URL : `https://www.numpy.org` (visité le 04/08/2022).

13. CHAGUÉ, CLÉRICE et ROMARY, « HTR-United ».

| | | | | |
|---|---|---|---|---|
| 255 | 255 | 251.67 | 85 | 251.67 |
| 255 | 255 | 135 | 135 | 85 |
| 255 | 251.67 | 85 | 218.33 | 85 |
| 255 | 85 | 135 | 251.67 | 135 |
| 251.67 | 85 | 218.33 | 255 | 218.33 |
| 168.33 | 135 | 251.67 | 255 | 255 |
| 85 | 218.33 | 255 | 255 | 255 |

(a) La moyenne de chaque pixel

| | | | | |
|---|---|---|---|---|
| 1.00 | 1.00 | 0.98 | 0.33 | 1.98 |
| 1.00 | 1.00 | 0.53 | 0.53 | 0.33 |
| 1.00 | 0.98 | 0.33 | 0.86 | 0.33 |
| 1.00 | 0.33 | 0.53 | 0.98 | 0.53 |
| 0.98 | 0.33 | 0.86 | 1.00 | 0.86 |
| 0.66 | 0.53 | 0.98 | 1.00 | 1.00 |
| 0.33 | 0.86 | 1.00 | 1.00 | 1.00 |

(b) L'image en niveau de gris

| | | | | |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 |

(c) L'image binarisée

FIGURE 3.5 – La binarisation

à droit de l'espace et les reconnaîtrait comme étant une ligne de texte. Le terme « masque » signifie l'ensemble des pixels qui contient une ligne de texte ou un caractère. [14] Du point de vue du système informatique, un masque est un ensemble de coordonnées signifiant un objet contigu qui encadre une ligne de texte ou un caractère. Cet objet contigu est la base à partir de laquelle le modèle HTR peut reconnaître le texte dedans.

### La transcription

La transcription du texte est l'étape fondamentale de l'HTR. Elle s'effectue à partir des données encadrées dans les masques et en appliquant un modèle de reconnaissance,

14. Denis COQUENET, Clément CHATELAIN et Thierry PAQUET. « Handwritten Text Recognition : From Isolated Text Lines to Whole Documents ». In : *ORASIS 2021*. Saint Ferréol, France : Centre National de la Recherche Scientifique [CNRS], sept. 2021. URL : `https://hal.archives-ouvertes.fr/hal-03339648` (visité le 04/08/2022).

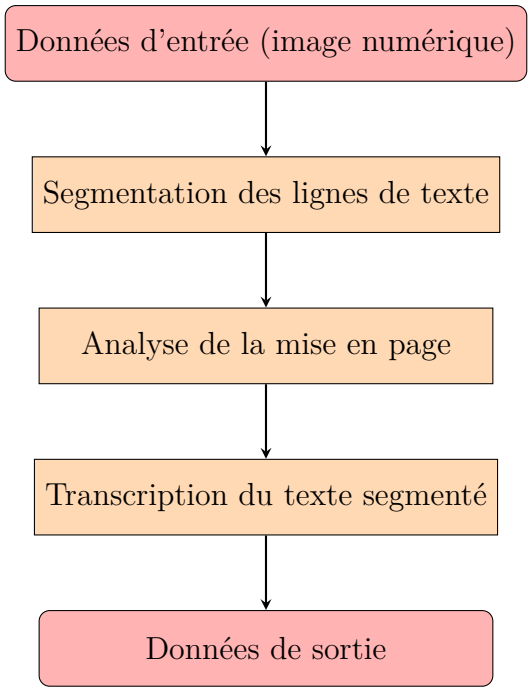



FIGURE 3.6 – Processus d'un logiciel HTR

parfois appelé le modèle HTR. [15]. La transcription du texte consiste à faire produire, par le modèle, une suite de caractères à partir de l'extrait de l'image qui lui est fourni en entrée et encadrée dans un objet « masque ». Pour résumer les deux étapes décrites ci-dessus, la sortie d'un logiciel HTR peut se constituer soit simplement du texte brut, soit d'un fichier hiérarchisé où le texte prédit et les données de la segmentation sont alignés. En tout cas, la phase de transcription est essentielle dans un protocole de reconnaissance automatique du texte.

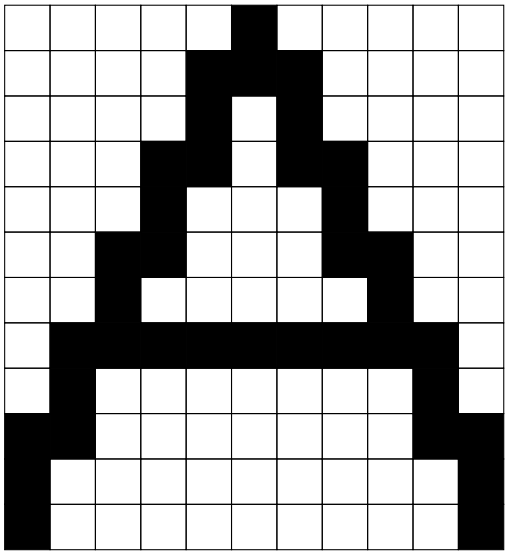
### 3.2.4 L'algèbre linéaire, les matrices, et l'intelligence artificielle

On transforme les pixels d'un caractère en une matrice, telle que celle présentée dans la figure 3.7b. Pourquoi ? À la base, l'ordinateur est une calculatrice. Les logiciels OCR et HTR décomposent une image numérique en représentations mathématiques, les matrices. Ainsi, les matrices permettent aux logiciels de faire des calculs probabilistes et de prédire quel caractère correspond à quelle représentation. Mais pour faire des prédictions, un logiciel a besoin d'une intelligence artificielle.

#### *Template matching*

Pendant la dernière moitié du XXe siècle, la reconnaissance du texte et l'intelligence artificielle ont connu des progrès importants. En 1950, il y a plus que soixante-dix ans,

15. L'acronyme HTR peut soit renvoyer uniquement à cette étape, soit plus généralement au processus complet qui comprend la reconnaîssance du texte, mais aussi la segmentation des zones et des lignes



(a) Le masque de la lettre $A$,
sur l'image binarisée

$$\begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}$$

(b) La matrice du masque de
la lettre $A$

FIGURE 3.7 – La matrice d'un caractère

Alan Turing a publié sa théorisation de l'IA.[16] Pendant les années soixante, au début de l'OCR, l'IA a rendu possible la prédiction du texte représenté par les matrices en faisant le lien entre deux entités : (1) la représentation qu'un logiciel a reconnue et (2) la représentation d'un caractère qu'il avait dans sa base de données. Cette technique initiale est le *template matching* ou le filtrage par modèle. Elle n'est pas le moyen le plus pratique. Les chercheurs V. K. Govindan et A. P. Shivaprasad l'ont revu en 1990, après qu'elle a

16. A. M. TURING. « Computing Machinery and Intelligence ». In : *Mind* LIX.236 (1er oct. 1950), p. 433-460. ISSN : 0026-4423. DOI : 10.1093/mind/LIX.236.433. URL : https://doi.org/10.1093/mind/LIX.236.433 (visité le 04/08/2022).

été dépassée par le *feature analysis*.

> *[Template matching] directly compares an input character to a standard set of prototypes stored [modèles stockés]. The prototype that matches most closely provides recognition. […] This type of technique suffers from sensitivity to noise and is not adaptive to differences in writing style.* [17]

Sous la dépendance du *template matching*, un logiciel OCR ne parviendra pas à préduire un caractère si la totalité de sa représentation en matrice n'est pas assez proche du modèle stocké lors de sa programmation.

#### *Feature analysis*

Les progrès dans le domaine de l'IA pendant les années 1970 et 1980 ont rendu possible le développement d'une nouvelle technique dont profite toujours les logiciels OCR, bien qu'elles aient été améliorées depuis. Au lieu de faire correspondre la représentation d'un caractère à une autre, les logiciels OCR décomposent un caractère en ses *features* ou ses aspects. En 1990, Govindan et Shivaprasad ont résumé l'état de cette technique.

> *The features may represent global and local properties of the characters. These include strokes, and bays in various directions, end points, intersections of line segments, loops […], and stroke relations, angular properties, sharp protrusions […]. These features have high tolerances to distortions and style variations, and also tolerate a certain degree of translation and rotation. However, the extraction processes are in general very complex and it is difficult to generate masks for these types of features.* [18]

Ainsi, le logiciel OCR essaie de faire correspondre l'ensemble de certains aspects d'une représentation à l'ensemble des mêmes aspects auxquels le logiciel associe un caractère. Selon Govindan et Shivaprasad, un aspect peut être un trait, le croisement des lignes, une boucle, la relation entre plusieurs traits, la caractéristique des angles, ou une saillie.

Grâce à la technique de *feature analysis*, un logiciel OCR moderne parvient à prendre une décision quant à la représentation d'un caractère bien qu'il n'ait pas trouvé le caractère de la même police, ayant la même représentation, dans sa base de données. *L'Handwritten Text Recognition* utilise aussi du *feature analysis*. Mais au lieu de produire les métadonnées sur la police du caractère ainsi que la langue du texte, l'HTR analyse les aspects d'un caractère simplement pour prédire le texte. Cependant, cette analyse n'est pas plus simple que celle réalisée par un logiciel OCR.

Le logiciel HTR a besoin d'associer beaucoup d'aspects d'une variété immense à un seul caractère. Une même main peut écrire une même lettre de plusieurs façons, qui

17. V. K. Govindan et A. P. Shivaprasad. « Character Recognition — A Review ». In : *Pattern Recognition* 23.7 (1990), p. 671. issn : 0031-3203. url : `https://www.academia.edu/6986960/Character_recognition_A_review` (visité le 05/08/2022).

18. Ibid.

produit une variation « intra-classe ». [19] En outre, l'écriture d'une main peut varier et une page peut avoir plusieurs mains. Dit simplement, l'OCR analyse les aspects d'un caractère du point de vue d'une langue et d'une police attendue. L'HTR se focalise uniquement sur les aspects topologiques d'un caractère, sans avoir besoin d'identifier la langue ni d'avoir appris la police du texte.

#### *Deep learning*

Aujourd'hui l'HTR profite du *feature analysis* ainsi que d'un développement plus récent dans l'intelligence artificielle qui s'appelle le *deep learning* ou l'apprentissage profond. Pouvant s'améliorer grâce aux réseaux neurones, un logiciel OCR ou HTR moderne peut prendre des décisions de plus en plus pertinentes quand il essaie de prédire du texte à partir de l'analyse des aspects (*feature analysis*). L'une des premières mises en œuvre des réseaux neurones pour *l'Handwritten Text Recognition* était le projet européen *Transkribus* qui a aussi mis au point l'OCR. [20]. Un autre projet européen a suivi *Transkribus* et, contrairement au premier, laisse ses données et les architectures de ses modèles ouvertes : *Kraken.* [21] Élaboré dans l'esprit de la science ouverte, le projet *Gallic(orpor)a* a choisi d'utiliser *Kraken* et une interface de transcription, ouverte elle aussi, *eScriptorium.*

## 3.3 Le modèle HTR

Puisque *Transkribus* et *Kraken* profitent tous les deux de l'apprentissage profond, les processus mis en œuvre par les interfaces graphiques *Transkribus* et *eScriptorium* ressemblent généralement à la même description. Ils commencent avec l'entraînement d'un modèle HTR. Ensuite, ils appliquent le modèle entraîné aux données d'entrées, c'est-à-dire l'image d'un page numérisée. Le le taux de réussite du processus est calculé en fonction du pourcentage des caractères que le modèle n'a jamais vus mais qui étaient bien prédits.

### 3.3.1 Les données d'entrée

Dans un premier temps, avant de penser à l'entraînement d'un modèle, il faut bien connaître la saisie des données. Les données d'entrée d'un modèle vont être les images numériques, composées des pixels. En général, leurs contenus textuels devraient être si-

19. Une même lettre écrite de la même manière par plusieurs mains produit une variation « inter-classe ».

20. Guenter Muehlberger et al. « Transforming Scholarship in the Archives through Handwritten Text Recognition : Transkribus as a Case Study ». In : *Journal of Documentation* 75.5 (9 sept. 2019), p. 954-976. ISSN : 0022-0418. DOI : 10.1108/JD-07-2018-0114. URL : https://www.emerald.com/insight/content/doi/10.1108/JD-07-2018-0114/full/html (visité le 04/08/2022).

21. Benjamin Kiessling. « Kraken - an Universal Text Recognizer for the Humanities ». In : ADHO 2019 - Utrecht. 2019. URL : https://dh-abstracts.library.cmu.edu/works/9912 (visité le 10/08/2022).

milaires afin que le modèle se spécialise dans une écriture particulière. Il est possible d'entraîner un modèle très généraliste, mais cela nécessite alors un très large corpus d'entraînement.

### 3.3.2 Les données d'entraînement

Le premier défi de l'entraînement d'un modèle HTR est la création des données d'entraînement. Ces données sont des transcriptions annotées et corrigées à la main à partir des images. L'ensembles des paires formées par une image et sa transcription s'appelle une vérité de terrain ou *ground truth* en anglais, puisque les transcriptions doivent être parfaites ou *vraies*[22]. Afin d'entraîner un modèle HTR, il faut un jeu des vérités de terrain. Alix Chagué décrit les vérités de terrain comme,

> « des ensembles de données annotées et corrigées de manière à fournir au modèle des paires composées d'une part d'une image ou d'une portion d'image (entrée) et d'autre part de l'annotation attendue (sortie), qui peut être des coordonnées dans le cas de la segmentation ou un ensemble de caractères pour la transcription. » [23]

Lors de l'apprentissage, les vérités de terrain fournissent au modèle fournissent au modèle le texte attendu à partir d'un segment d'image pour qu'il puisse savoir comment s'améliorer afin de produire les bonnes prédictions ou les prédictions *vraies*. Les données générées reproduisent au plus près la prédiction idéale des vérités de terrain. Grâce à l'apprentissage profond, un modèle peut apprendre comment arriver à la prédiction souhaitée à partir de données d'entraînement.

### 3.3.3 L'entraînement

Lors de l'entraînement, le modèle HTR (comme un modèle TAL)) s'évalue périodiquement et une dernière fois quand l'entraînement est terminié. La dernière évaluation s'appelle le *score*. Afin d'obtenir un meilleur *score*, on peut optimiser l'entraînement du modèle en jouant sur plusieurs paramètres.

Par exemple, on peut modifier le nombre de fois que le modèle révise sa manière de faire ses tâches de prédiction. Chaque essaie s'appelle une époque, dérivé de l'anglais *epoch*, et un plus grand nombre d'époque donne au modèle plus de chances de s'améliorer. Cependant, plus d'époque pèse plus sur le budget d'un projet puisqu'il consomme plus de puissance de calcul et prend plus de temps. En outre, on ne veut pas faire passer trop

22. David Lassner et al. « Publishing an OCR Ground Truth Data Set for Reuse in an Unclear Copyright Setting. Two Case Studies with Legal and Technical Solutions to Enable a Collective OCR Ground Truth Data Set Effort ». Version 1.0. In : *Fabrikation von Erkenntnis – Experimente in den Digital Humanities. Hg. von Manuel Burghardt* Lisa Dieckmann (2021). Avec la coll. d'Herzog August Bibliothek, 5). doi : 10.17175/SB005_006. url : https://zfdg.de/sb005_006 (visité le 10/08/2022).

23. Chagué, Clérice et Romary, « HTR-United ».

d'époque et risquer le sur-apprentissage d'un modèle, qui s'appelle le « sur-apprentissage » (*overfitting* en anglais). Cela veut dire que la fonction prédictive du modèle s'est trop bien adaptée à ses données d'entraînement, y compris tout le bruit des données, et elle n'est pas suffisamment généraliste pour réussir sur des données que le modèle n'aurait jamais vues. On peut également régler la taille du pas d'apprentissage après chaque époque, c'est à dire son *learning rate*.

On peut aussi modifier la composition du jeu de données traité lors d'une époque. Ce dernier paramètre s'appelle un *batch size*, et il est aussi un entier, comme l'époque. Disons qu'on veut entraîner notre modèle sur 400 images. Une époque prendrait trop de temps et trop de puissance de calcul s'il essayait de traiter toutes les images de notre jeu de données au même temps. Du coup, on veut le diviser selon notre paramètre *batch size*. Si on déclarait un *batch size* de 100 images, on dirait au modèle qu'il faut itérer sur son jeu de données 4 fois afin de compléter une époque, en rappelant qu'une époque est égale à une passe complète sur le jeu de données d'entraînement, voir la figure 3.8.

Ce qu'on appelle l'erreur, soit la différence entre la prédiction du modèle et la vérité de terrain, se calcule à chaque itération d'un *batch* dans une époque. On veut que cette valeur diminue, c'est à dire que la prédiction du modèle se rapproche de plus en plus à la vérité de terrain. Pour optimiser le modèle, on peut choisir entre plusieurs fonctions pour calculer l'erreur, ce qu'on appelle une *loss function*. Dans l'exemple de la Figure 3.8, on pourrait appliquer une *loss function*, telle que le *mean squared error* (MSE), à toutes les prédiction d'un *batch* afin de connaître l'erreur ou le *loss* du modèle à ce point de l'entraînement.

Dans le cadre du projet *Gallic(orpor)a*, Ariane Pinche a entraîné des modèles HTR sur le corpus *gold*, que l'équipe a fait à la main et qu'elle et Simon Gabay ont vérifié. Pinche l'a scindé en trois, comme la Figure 3.9 visualise. Une majorité des images (80%) compose le jeu d'entraînement (*training set*). Une petite partie (10%) compose le jeu de validation (*development set*). Et la dernière partie (10%) compose le jeu de test (*testing set*). Les données du *training set* sont ceux qui se divisent en *batches* ; chacune est traitée une fois lors d'une époque. En prenant compte du taux de réussite du modèle sur les données du *training set*, le réseau neuronal fait des ajustements à la nature des connexions entre ses neurones, spécifiquement comment chaque connexion pèse sur le réseau. Après quelques ajustements, l'entraînement s'arrête pour évaluer le modèle en cours de développement. Les données du *development set* sont utilisées pour donner au réseau une évaluation impartiale de son succès puisqu'elles sont des données qu'il n'a pas pris en compte lors de l'ajustement des poids des connexions. Enfin, les données du *testing set* sont utilisées à la fin de l'entraînement pour donner un *score* final au modèle sur les données qu'il n'a jamais vues.

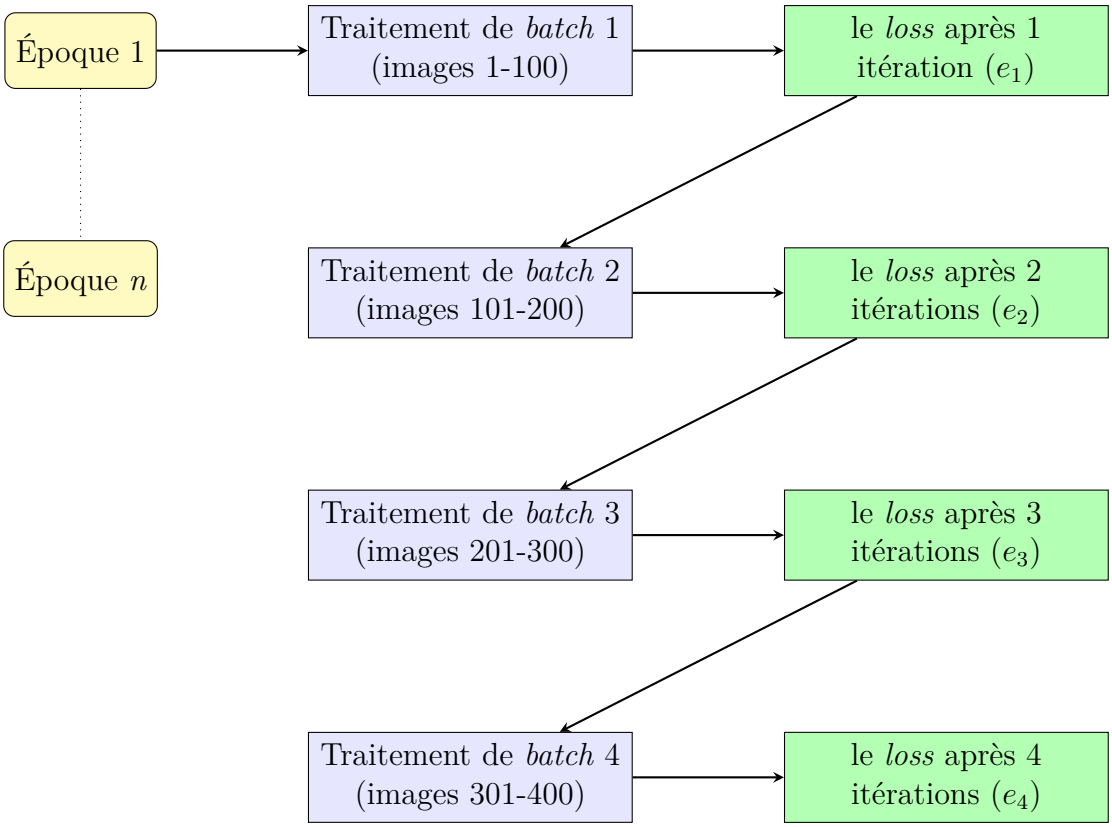



FIGURE 3.8 – La visualisation d'époque 1


FIGURE 3.9 – La division du corpus gold pour l'entraînement d'un modèle

### 3.3.4 Le résultat de l'entraînement

Le résultat de l'entraînement s'appelle un modèle. L'un des buts du projet *Gallic(orpor)a* était de proposer pour tous les documents du XVe au XIXe siècle deux modèles généralistes. Un se spécialisera dans l'écriture des manuscrits et des incunables et l'autre sur les imprimés.

# Deuxième partie

# Exposition de la préparation et du travail d'analyse

# Chapitre 4

# Un pipeline visant à tout rassembler

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.

## 4.1 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.

### 4.1.1 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.

#### 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 4.1, qui montre l'exemple de deux documents sources dans lequel chacun a trois pages numérisées en format JPEG. 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.

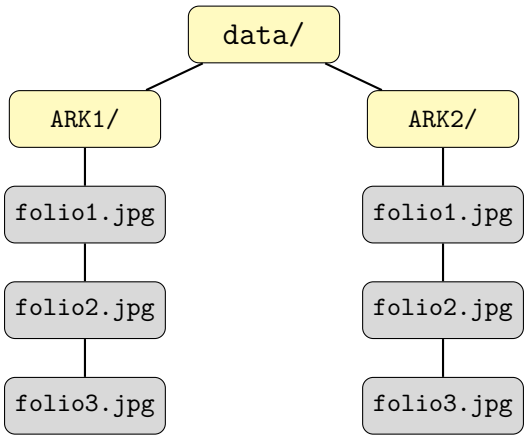



FIGURE 4.1 – Système de fichiers

### 4.1.2 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

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

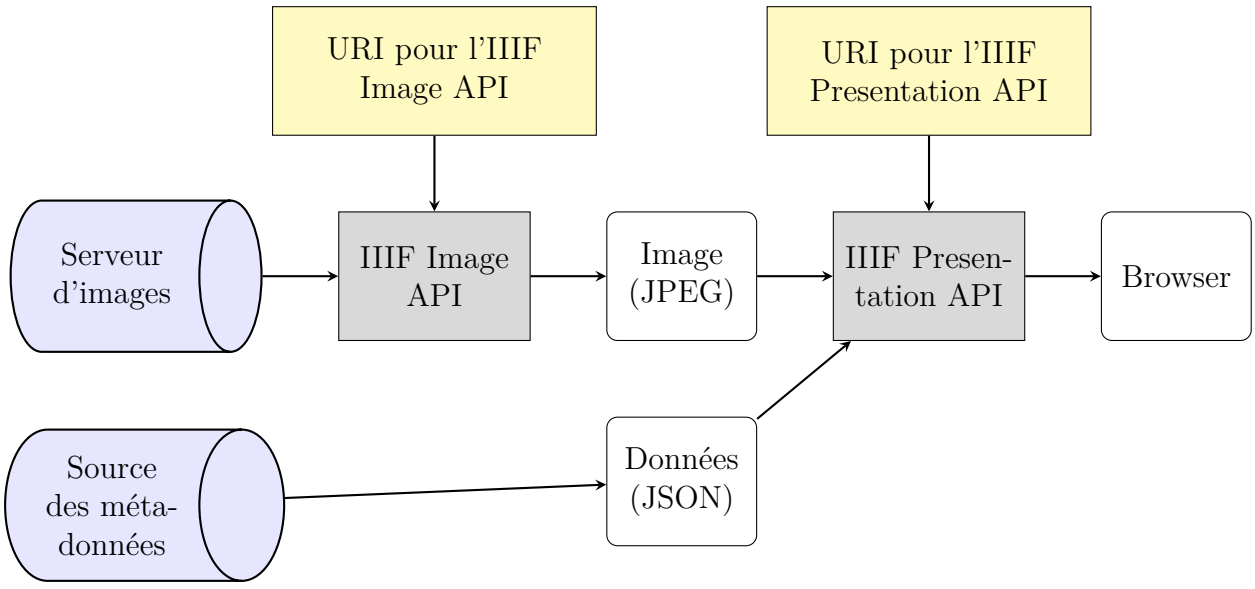
la gestion des ressources visuelles mises en ligne avec le but d'améliorer l'exploitation et la partage des ressources et de leurs métadonnées. Suite à une requête standardisée HTTP(S), l'IIIF Image API renvoie l'image. L'URI se compose de quatre éléments généralisés, suivis par cinq éléments qui peut 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 4.2 – IIIF APIs

La Figure 4.2 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 4.2a. Comme

la Figure 4.2c 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 4.2b, qui renverrait des données en format JSON.

### 4.1.3 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.

## 4.2 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.

### 4.2.1 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

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).

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.

#### 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 3. 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.

### 4.2.2 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 4.2c, 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.

### 4.2.3 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.

### 4.2.4 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 4.3. 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.

Le schéma XML ALTO peut se réaliser dans plusieurs formats, et pas uniquement celui qui se voit dans la Figure 4.3. Dans l'exemple de la Figure 4.3, les données textuelles

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.

```
<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>
```

FIGURE 4.3 – L'encodage d'une ligne de texte en ALTO

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 4.3, 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.

## 4.3 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 a 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 le 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 les données dans une façon hiérarchisée. Mais contrairement 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 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.

### 4.3.1 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 définissent 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 5.

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

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).

### 4.3.2 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 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.

### 4.3.3 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>`.

## 4.4 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.

### 4.4.1 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 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.

## 4.5 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).

9. Bawden et al., « Automatic Normalisation of Early Modern French ».

# Chapitre 5

# L'analyse des structures des données XML

## 5.1 ALTO : *Analyzed Layout and Text Object*

### 5.1.1 Qu'est-ce qu'est l'ALTO ?

Le format XML ALTO s'est évolué à partir du projet européen METAe en 2003. Le projet qui a conditionné la création d'ALTO s'occupait du développement des logiciels dont des institutions patrimoniales pourraient servir à la création et à l'exploitation des fac-similés numériques de leur fonds. Le but du projet était d'extraire à partir des pages numérisées les informations portant sur la mise en page et les autres données structurelles. D'ici 2003, les logiciels OCR étaient déjà bien mis en pratique. L'enjeu à l'époque était d'élaborer un schème de données qui soumettrait le texte extrait à la logique structurelle de la page et du document. Tandis qu'un logiciel OCR reconnaît le texte du titre d'un chapitre et le texte de son sous-titre, un nouveau format devrait pouvoir distinguer les deux lignes de texte que le logiciel a reconnues et ensuite les hiérarchiser, en disant que le sous-titre est subordonné au titre du chapitre.

Le projet METAe a donc développé un format METS (*Metadata Encoding and Transmission Standard*) qui avait pour but d'augmenter les données textuelles extraites par un logiciel OCR avec la logique de la mise en page et du document. Bien que les logiciels OCR aient souvent exporté leurs prédictions dans un format du texte brut, le format METS visait à hiérarchiser le mélange des diverses données dans un format XML. Par exemple, à travers d'un arbre XML, le texte d'un sous-titre descendrait de la région dans laquelle les caractères de cette ligne de texte s'encadrent sur la page numérisée. En outre, au moins l'un d'eux, soit la donnée sur le texte du sous-titre, soit les données sur l'emplacement, porterait quelque chose pour dire que dans la logique du document il est un sous-titre.

Le schème METS a réussi à combiner les métadonnées de la ressource numérique

ainsi que de l'objet text qui a été transcrit avec ses données structurelles et textuelles grâce au format hiérarchisé de l'XML. Mais le format primordial METS n'avait pas répondu à la question de comment bien structurer les dernières données, celles qui sont produites par un logiciel OCR ou HTR. Les créateurs du format ALTO ont décrit son prédécesseur comme un « emballage » (*wrapper*) pour la structure de données ALTO. [1] Tandis que le format METS organise les métadonnées et la logique du document, telles que l'occurence et l'ordre des pages, le format ALTO s'insère sous l'arbre de chaque page afin de décrire la transcription produite pour l'image numérique.

Normalement, un fichier XML ALTO décrit une page (ou une image) d'un document. Mais, comme se voit dans l'exemple donné dans l'exposition du schème quand il était nouveau en 2003, modélisé dans la Figure 5.1, l'élément `<Layout>` peut en fait contenir plusieurs éléments `<Page>`. [2] Néanmoins, la plupart de logiciels HTR qui utilisent le format ALTO crée un fichier per page numérisée.

```
<Layout>
  <Page ID="XXX" PHYSICAL_IMG_NR="000" HEIGHT="000" WIDTH="000" STYLEREFS="
    XXX">
    <PrintSpace ID="XXX" HPOS="000" VPOS="000" HEIGHT="000" WIDTH="000">
      <TextBlock ID="XXX" HPOS="000" VPOS="000" HEIGHT="000" WIDTH="000">
        <TextLine ID="XXX" HPOS="000" VPOS="000" HEIGHT="000" WIDTH="000">
          <String ID="XXX" HPOS="000" VPOS="000" HEIGHT="000" WIDTH="000"
    CONTENT="XXX"/>
          <Sp ID="XXX" HPOS="000" VPOS="000" HEIGHT="000" WIDTH="000"/>
          <String ID="XXX" HPOS="000" VPOS="000" HEIGHT="000" WIDTH="000"
    CONTENT="XXX"/>
        </TextLine>
        <TextLine ID="XXX" HPOS="000" VPOS="000" HEIGHT="000" WIDTH="000">
          <String ID="XXX" HPOS="000" VPOS="000" HEIGHT="000" WIDTH="000"
    CONTENT="XXX"/>
          <Sp ID="XXX" HPOS="000" VPOS="000" HEIGHT="000" WIDTH="000"/>
          <String ID="XXX" HPOS="000" VPOS="000" HEIGHT="000" WIDTH="000"
    CONTENT="XXX"/>
        </TextLine>
      </TextBlock>
    </PrintSpace>
  </Page>
  <Page ID="XXX" PHYSICAL_IMG_NR="000" HEIGHT="000" WIDTH="000" STYLEREFS="
    XXX">
    <PrintSpace ID="XXX" HPOS="000" VPOS="000" HEIGHT="000" WIDTH="000">
<!-- ... -->
</Layout>
```

FIGURE 5.1 – La structure ALTO version 1, circa 2003

Le format ALTO est dans sa quatrième version, mais la structure actuelle ressemble

1. Birgit STEHNO, Alexander EGGER et Gregor RETTI. « METAe–Automated Encoding of Digitized Texts ». In : *Literary and Linguistic Computing* 18.1 (1er avr. 2003), p. 77-88. ISSN : 0268-1145, 1477-4615. DOI : 10.1093/llc/18.1.77. URL : https://academic.oup.com/dsh/article-lookup/doi/10.1093/llc/18.1.77 (visité le 24/08/2022), p. 81.
2. Ibid.

bien au modèle qu'ont présenté les auteurs Birgit Stephno, Alexader Egger, et Gregor Retti en 2003. Dans sa première version, montrée dans la Figure 5.1, les éléments les plus petits étaient les segments de texte (`<String>`) et les espaces entre mots (`<Sp>`), balisés dans une ligne de texte (`<TextLine>`) qui appartient à un bloque de texte (`<TextBlock>`). Tous ces éléments XML porte un identifiant unique (`@ID`) et quatre coordonnées portant sur le rectangle dans lequel s'encadre le contenu de l'élément. Le contenu textuel est représenté dans l'attribut `@CONTENT` de l'élément `<String>`.

### 5.1.2 La structure actuelle des fichiers XML ALTO

Aujourd'hui, l'élément le plus petit d'une structure de données ALTO est un glyphe (`<Glyph>`), au lieu d'un segment de caractères (`<String>`). Par conséquent, dans le nouveau format, le contenu textuel est représenté deux fois, une fois comme l'attribut `@CONTENT` de l'élément classique `<String>` et une deuxième comme le même attribut de l'élément `<Glyph>`. Un comparaison entre les deux arborescences est visualisé dans la Figure 5.3. Comme montre la sous-figure 5.3b, la nouvelle architecture se permet d'aller en plus de détail. Certains logiciels, tel que l'interface *eScriptorium*, produisent toujours les fichiers ALTO avec une variation de l'ancienne structure où l'élément `<String>` n'est pas répétable et représente le contenu textuel de la ligne.

En général, toute donnée portant sur la mise en place de la page se dispose de quatre coordonnées qui ensemble tracent un rectangle. Les valeurs des attributs `@HPOS` et `@VPOS` font les coordonnées x,y du point le plus haut à gauche du rectangle, comme se voit dans la Figure 5.2. La valeur de l'attribut `@HEIGHT` compte la différence entre la coordonnée y du point le plus haut et la coordonnée y du point le plus bas. La valeur de l'attribut `@WIDTH` calcule aussi la différence entre le côté gauche du carré et son côté droit. Ces quatre attributs sont attribués aux éléments `<PrintSpace>`, `<TextBlock>`, `<TextLine>`, `<String>`, `<Sp>`, et `<Glyph>`.

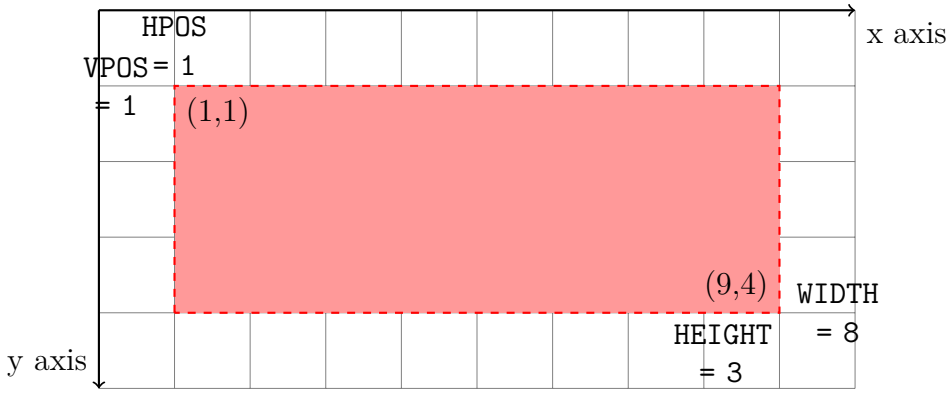
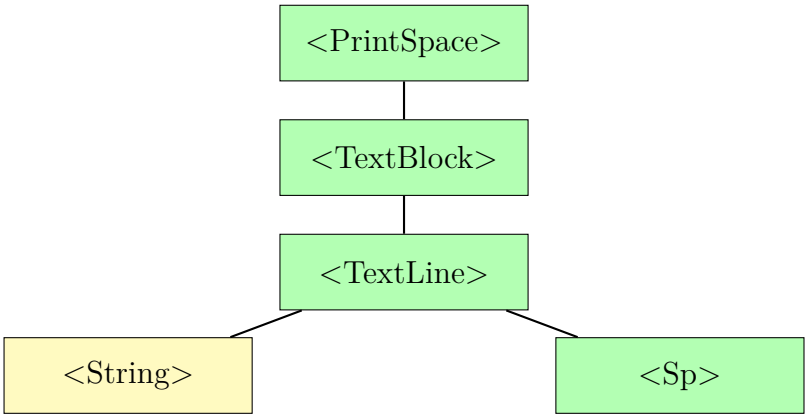


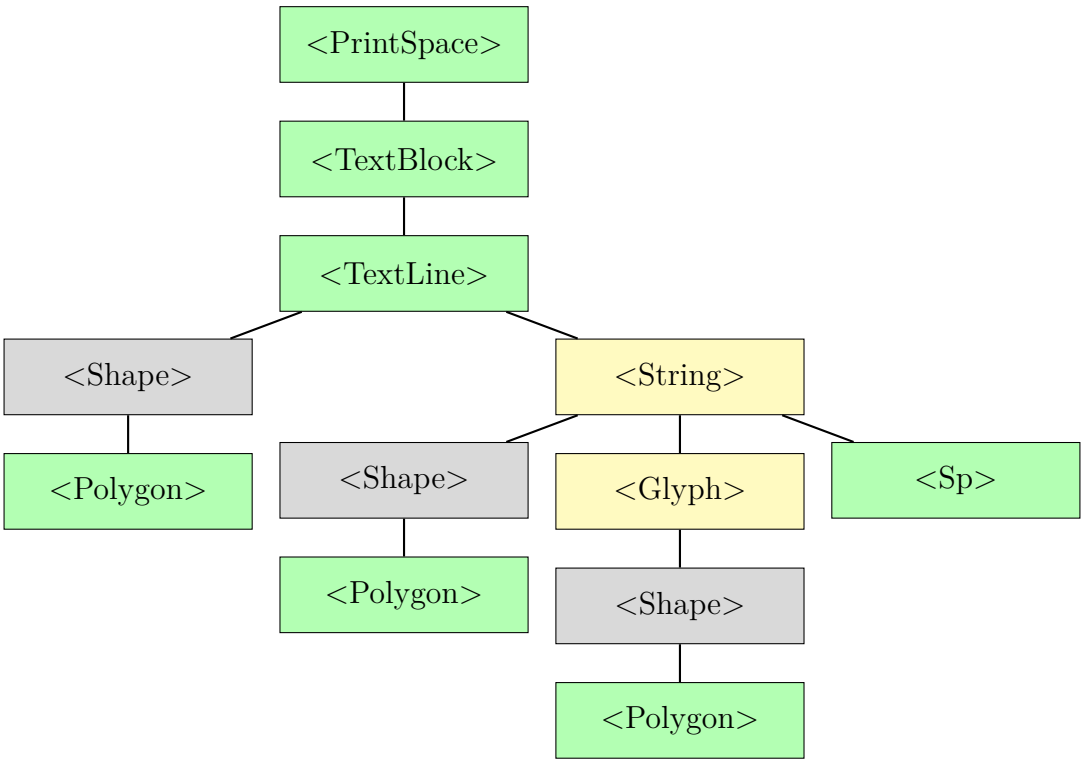

FIGURE 5.2 – Les coordonnées d'un masque en rectangle

L'arborescence actuelle du format ALTO diffère de l'original car, aujourd'hui, elle peut préciser les coordonnées d'un polygone en plus d'un rectangle. Cela est un déve-

loppement dans la technologie des logiciels OCR et HTR. La visualisation dans la sous-figure 5.3b montre le nouvel élément `<Polygon>` qui descend directement d'un élément (`<Shape>`) qui lui-même ne porte pas d'attribut ni d'intérêt dans l'arborescence que de baliser les informations du polygone. Cet élément est indiqué en gris dans la sous-figure 5.3b. La Figure 5.3 indique tout élément qui contient du texte en jaune dans les deux arborescences. Le contenu textuel est toujours balisé dans l'élément `<String>`, qui porte sur le segment ou sur le mot d'une ligne de texte (`<TextLine>`). Mais en allant jusqu'au détail du glyphe dans l'arborescence actuelle, l'élément `<Glyph>` représente tout caractère composant un mot (`<String>`).




(a) Ancienne structure




(b) Nouvelle structure

FIGURE 5.3 – Modélisation des formats ALTO

Certains attributs actuels dans l'arborescence, tel que l'attribut `@BASELINE` de l'élément `<TextLine>` et l'attribut `@POINTS` de tout élément `<Polygon>`, prennent comme valeur une chaîne d'entiers. Montrée dans la Figure 5.4, cette chaîne veut représenter des pairs de coordonnées x,y. Chaque point articule une extrémité soit d'une ligne qui trace le baseline de la ligne de texte (`@BASELINE`) soit une ligne qui encadre la région reconnue par un modèle de segmentation (`@POINTS`). Pour les deux attributs `<@POINTS>` et `<@BASELINE>`, le format ALTO encode chaque entier dans une chaîne dont les composants sont séparés par espace, où la valeur de l'axe des x précède la valeur de l'axe des y. Un polygone (`@POINTS`) peut avoir plusieurs points tout le long de son périmètre. Par contre, le baseline d'une ligne de texte (`@BASELINE`) compte toujours quatre entiers puisqu'il n'a qu'un début et une fin, donc deux pairs x,y.

```
<Layout>
  <Page ID="XXX" PHYSICAL_IMG_NR="000" WIDTH="000" HEIGHT="000">
    <PrintSpace HPOS="000" VPOS="000" HEIGHT="000" WIDTH="000">
      <TextBlock ID="XXX">
        <TextLine ID="XXX" HPOS="000" VPOS="000" HEIGHT="000" WIDTH="000"
    BASELINE="1 2 3 4">
          <Shape>
            <Polygon POINTS="1 2 3 4 5 6 7 8"/>
          </Shape>
          <String ID="XXX" CONTENT="AB" HPOS="000" VPOS="000" WIDTH="000"
    HEIGHT="000" WC="1.0">
            <Shape>
              <Polygon POINTS="1 2 3 4 5 6 7 8 9 10"/>
            </Shape>
            <Glyph ID="XXX" CONTENT="A" HPOS="000" VPOS="000" WIDTH="000"
    HEIGHT="000" GC="1.0">
              <Shape>
                <Polygon POINTS="1 2 3 4 5 6 7 8"/>
              </Shape>
            </Glyph>
            <Glyph ID="XXX" CONTENT="B" HPOS="000" VPOS="000" WIDTH="000"
    HEIGHT="000" GC="1.0">
              <Shape>
                <Polygon POINTS="1 2 3 4 5 6 7 8"/>
              </Shape>
            </Glyph>
          </String>
        </TextLine>
<!-- ... -->
      </TextBlock>
    </PrintSpace>
  </Page>
</Layout>
```

FIGURE 5.4 – La structure ALTO version 4, circa 2022

## 5.2 TEI : *Text Encoding Initiative*

La raison pour laquelle la TEI a été choisie pour fusionner toute donnée du pipeline *Gallic(orpor)a* est parce qu'elle est un format XML souple, qui peut s'adapter facilement à plusieurs types de document et d'édition numérique. Cela veut dire qu'une exposition détaillée du schème TEI n'est pas possible. La manière pour encoder une ligne de texte n'est pas aussi fixée que cela du schème ALTO, par exemple.

Tandis qu'une ligne de texte dans un fichier ALTO est balisée dans l'élément `<TextLine>`, une ligne de texte dans un document TEI peut être encodée dans plusieurs façons. Elle peut suivre l'élément vide `<lb/>` ou elle peut être à côté d'autres lignes de texte, toutes balisées ensemble dans un élément tel que `<p>` ou `<div>`. De plus, parce que la TEI permet de classer les composants d'une ligne de texte selon la logique du document ou de la langue, certains mots ou phrases peuvent être balisés dans d'autres éléments, tel que l'élément `<date>` pour encoder une année. Dans le même ordre des idées, une ligne de texte peut aussi être balisée dans les éléments qui expliquent sa fonction dans le document. Par exemple, une ligne de texte peut être un item dans une liste (`<item>`) ou la salutation à la fin d'une lettre (`<salute>`).

Prenant l'exemple d'une lettre, la Figure 5.5 montre l'encodage de sa salutation dans les deux schèmes, ALTO et TEI. On voit que l'ALTO excelle à préciser l'emplacement des mots (et des caractères) sur la page d'un document. Mais après la reconnaissance de la lettre, dont la certitude du modèle se représente par l'attribut `@WC` (*word certainty*), le schème ALTO ne donne pas d'autre information. L'encodage dans le format TEI, par contre, enrichit la ligne de texte avec beaucoup d'information. Grâce à l'élément TEI `<salute>` on sait que la ligne de texte est la salutation d'une lettre ou quelque autre forme de communication. De plus, l'encodage appuie sur la date en-tête pour attribuer au mot *demain* une date précise qui est encodée dans l'attribut `@when`. En fin, le schème TEI dispose d'un système pour réunir les occurrences du même concept dans un texte, tel qu'une personne. L'encodage dans la Figure 5.5c utilise l'attribut `@ref` pour dire que l'occurence du mot *chérie* fait référence à une personne à laquelle a été donnée, dans les métadonnées du document TEI, l'identifiant `"Kelly"`.[3]

### 5.2.1 Qu'est-ce qu'est la TEI ?

Comme montrent les exemples de la Figure 5.5, le schème TEI se spécialise à la représentation d'un texte et à son édition numérique. Il facilite l'enrichissement du texte avec les métadonnées, telles que les références aux autres endroits dans le document ainsi que la classification de la nature d'un mot ou d'une phrase. Les normes de la TEI sont souples à exprès, afin de permettre les encodages personnalisés qui se focalisent sur les

3. Dans le TEI, les identifiants n'ont pas de mot-dièse, mais quand ils sont référencés dans le document la référence en porte un.

12 août 2022

*Coucou ! J'ai fait une réservation*
*pour ton anniversaire.*

*À demain ma chérie,*

(a) L'exemple d'une lettre

```
<TextLine ID="line1" HPOS="0" VPOS="0" HEIGHT="40" WIDTH="200" BASELINE="0
    40 200 40">
  <Shape>
    <Polygon POINTS="...."/>
  </Shape>
  <String ID="seg1" CONTENT="À" HPOS="..." VPOS="..." WIDTH="..." HEIGHT="
    ..." WC="1.0">
  <!-- ... -->
  <String ID="seg2" CONTENT="demain" HPOS="..." VPOS="..." WIDTH="..."
    HEIGHT="..." WC="0.888">
  <!-- ... -->
  <String ID="seg3" CONTENT="ma" HPOS="..." VPOS="..." WIDTH="..." HEIGHT="
    ..." WC="0.9">
  <!-- ... -->
  <String ID="seg3" CONTENT="chérie," HPOS="..." VPOS="..." WIDTH="..."
    HEIGHT="..." WC="0.91">
  <!-- ... -->
</TextLine>
```

(b) La dernière ligne de la lettre encodée dans le schème ALTO

```
<salute>À <date when="2022-08-13">demain</date> ma <name ref="#Kelly" type=
    "person">chérie</name>,</salute>
```

(c) La dernière ligne de la lettre encodée dans le schème TEI

FIGURE 5.5 – Le comparaison de l'encodage d'une ligne de texte en ALTO et TEI

aspects différents d'un texte. Le même texte peut donc être encodé en TEI dans plusieurs manières, selon les besoins et les objectifs des personnes qui se chargent de l'encodage.

Les normes de la TEI sont maintenues par une communauté internationale et leur usage est très répandu dans le monde. Naomi Truan et Laurent Romary ont dit en 2021 que la TEI *has become, since its inception in 1987, the reference technical standard for the representation of textual content in the humanities.*[4] Aujourd'hui l'association est soutenue par le financement des institutions patrimoniales qui comptent sur ses *guidelines* et contribuent des cas d'utilisation. Sur son site web, l'association explique qu'elle continue à modifier ses normes selon les besoins des utilisateurs.

> The scope of the TEI is constantly expanding and the Guidelines are in steady ongoing development to keep pace with the emerging needs of the TEI community.[5]

La croissance de la TEI rend le schème très approprié à l'édition et à l'échange puisque beaucoup d'institutions ont développé des outils numériques qui l'utilisent.

La souplesse de la TEI est à la fois un avantage et un défi à surmonter. Puisque le schème permet de plusieurs encodages du même document, il est donc possible de réaliser plusieurs transformations d'un encodage en ALTO vers un encodage en TEI. Mais pour mettre en œuvre une transformation automatique à l'échelle, il faut une seule modélisation qui s'adapte à tout type de document dont la transcription est encodée en ALTO. En outre, l'enrichissement du texte possible dans la TEI est compliqué à réaliser par ordinateur. Tandis qu'un humain pourrait voir la date en-tête sur la lettre dans la Figure 5.5a et puis savoir que la date référencée dans la salutation est le jour suivant, le 13 août, un logiciel ne pourrait pas faire le liaison entre les deux données si facilement. Donc, bien qu'il puisse savoir, grâce au TAL, que le mot *demain* veut parler d'une date, il ne saurait pas de quelle date parle la lettre ; par contre, une lectrice ou un lecteur humain la saurait avec facilité. Voici quelques défis d'une transformation d'ALTO à TEI.

### 5.2.2 Les éléments de base de la TEI

La TEI peut s'adapter à plusieurs types de documents mais elle exige toujours certains éléments de la racine qui donnent au schème son arborescence générale. Depuis la racine `<TEI>` d'un document TEI, il faut au moins ces deux descendants : le `<teiHeader>` et le `<body>`. Comme le schème ALTO, le schème TEI a besoin des métadonnées à propos du document encodé et de l'encodage lui-même. Le document TEI imbrique les métadonnées dans l'élément `<teiHeader>`. L'élément `<body>` porte sur les données qui constituent

4. Naomi Truan et Laurent Romary. « Building, Encoding, and Annotating a Corpus of Parliamentary Debates in TEI XML : A Cross-Linguistic Account ». In : *Journal of the Text Encoding Initiative* Issue 14 (Issue 14 17 mars 2021). ISSN : 2162-5603. DOI : 10.4000/jtei.4164. URL : https://journals.openedition.org/jtei/4164#tocto2n4 (visité le 26/08/2022), p. 21.

5. *About – TEI : Text Encoding Initiative*. URL : https://tei-c.org/about/ (visité le 25/08/2022).

la transcription ou la représentation du document ou des documents ; le dernier sera le cas où le document TEI réalise une édition critique qui ressemblent plusieurs exemplaires d'une œuvre, par exemple. Pour résumer, la TEI a besoin d'au moins les métadonnées, encodées dans le `<teiHeader>`, et les données qui représentent le texte, encodées dans le `<body>`.

Après ces deux éléments obligatoires, le schème TEI autorise d'autres éléments facultatifs de descendre directement de la racine `<TEI>`. L'un d'eux est l'élément `<sourceDoc>` dont nous parlons dans la section 4.3.1. La TEI définit le `<sourceDoc>` comme un élément qui peut contenir *une transcription ou une représentation d'un seul document source, qui se réserve le pouvoir à faire partie d'un dossier génétique ou d'une collection d'autres sources.*[6] (traduction par l'autrice) Comme se justifie dans la section 4.3.1, le projet *Gallic(orpor)a* a choisi d'encoder toute donnée du fichier ALTO dans l'élément TEI `<sourceDoc>`. Le schème TEI destine l'élément `<sourceDoc>` à la transcription d'un document source. Un fichier ALTO contient une telle transcription, produite par un logiciel OCR ou HTR. Le `<sourceDoc>` convient bien aux données d'un fichier ALTO car les éléments qui descendent du `<sourceDoc>` portent sur la mise en page ainsi que sur la transcription des images de texte.

Un autre élément facultatif qui descend de la racine `<TEI>` est l'élément `<standOff>`. Le projet *Gallic(orpor)a* s'appuyait aussi sur cet élément parce qu'il est destiné à la représentation des annotations au texte. L'avant dernière étape du pipeline du projet est l'analyse linguistique du texte prédit par le logiciel HTR. Le résultat de cette analyse est une version du texte annotée qui pourrait se différer sensiblement da la transcription. Selon les *guidelines* de la TEI :

> The `standOff` element is intended to hold content that does not fit well in the `text` (e.g. because it is not transcribed from the source), nor in the `teiHeader` (e.g. because it is not metadata about the source or transcription). Examples include [...] annotations indicating the morphosyntactic features of a text, and annotations commenting on or associating parts of a text with additional information.[7]

Comme s'est révélé par les *guidelines*, l'élément `<standOff>` convient bien au texte annoté et normalisé. Ainsi, la transcription du texte tel qu'il s'apparaît dans le document source, avec toute saute de ligne et faute d'orthographe, se trouvera dans les éléments `<sourceDoc>` et `<text>`, qui descend du `<body>`. Par contre, la version du texte qui n'existe pas dans le document source mais qui sert bien à l'analyse du document se trouvera dans l'élément facultatif `<standOff>`. Pour résumer, les quatre éléments pertinents qui descendent de la racine `<TEI>` sont visualisés dans la Figure 5.6.

6. *TEI element sourceDoc.*

7. *16 Linking, Segmentation, and Alignment - The TEI Guidelines.* URL : `https://tei-c.org/release/doc/tei-p5-doc/en/html/SA.html#SASOstdf` (visité le 25/08/2022).

```xml
<TEI>
  <teiHeader><!-- métadonnées --></teiHeader>
  <sourceDoc><!-- transcription --></sourceDoc>
  <body>
    <text><!-- texte pré-éditorialisé --></text>
  </body>
  <standOff><!-- texte annoté --></standOff>
</TEI>
```

FIGURE 5.6 – Les éléments de base du schème TEI

# Chapitre 6

# À la recherche des métadonnées

Tout document TEI doit porter des métadonnées qui renseignent sur la ressource TEI elle-même et le texte qu'elle représente. Si le texte est la transcription d'un document source numérisé, qui est le cas pour tout document du projet *Gallic(orpor)a*, trois objets sont concernés. En plus de la ressource créée, il y a le document source physique qui a été numérisé ainsi que la numérisation elle-même qui a été traitée par les modèles HTR. Pour le projet *Gallic(orpor)a* qui avait pour but le traitement automatisé des fac-similés numériques, les métadonnées du document TEI portent sur les trois objets de texte suivants :

1. la ressource lexicographique numérique, c'est-à-dire le document TEI lui-même que le pipeline *Gallic(orpor)a* produit

2. la numérisation du document source, c'est-à-dire le fac-similé numérique stocké dans la base de données Gallica

3. le document source physique, qui appartient du fonds de la Bibliothèque nationale de France et a été numérisé

Ce chapitre explique de quoi consistent les métadonnées de ces trois objets de texte ainsi que le format par lequel elles sont accédées. Malgré la diversité de la portée du projet *Gallic(orpor)a*, chaque document traité concerne ces trois objets puisque chacun est un fac-similé numérique, dérivé d'un document source physique et transformé en ressource lexicographique par le pipeline. Pour être faites à l'échelle, les métadonnées doivent être les mêmes pour tout type de document. Un manuscrit écrit par plusieurs mains et apparu sans éditeur, issue d'un scriptorium à une date approximative, doit se disposer de mêmes types de métadonnées qu'un imprimé écrite par une autrice et publiée par un éditeur. Une solution pour surmonter ce défi est de minimiser les métadonnées portant sur le document source et d'encoder uniquement *l'essentiel*. Au sein de cette solution est la sélection d'en quoi constitue l'essentiel.

**La structure du `<teiHeader>`**

Les métadonnées de la ressource lexicographique du pipeline *Gallic(orpor)a* sont toutes encodées dans un élément XML que le schème TEI exige : le `<teiHeader>`. Dans le cadre du projet *Gallic(orpor)a*, le `<teiHeader>` servait à enrichir l'exploitation de la ressource numérique ainsi qu'à faciliter l'échange de ses données. Selon les normes de la TEI, trois parties peuvent constituer le `<teiHeader>` : une description bibliographique de l'encodage (`<fileDesc>`), une description des aspects non bibliographiques tel qu'un classement du contenu textuel qui appartiennent au texte représenté (`<profileDesc>`), et une description technique de l'encodage et/ou l'appareil qui l'a fait (`<encodingDesc>`).

## 6.1 La description bibliographique (`<fileDesc>`)

La première chose sur laquelle la ressource numérique doit renseigner est elle-même. Ces informations sont organisées dans l'élément `<fileDesc>`. Traduit littéralement en français comme *la description du fichier*, le `<fileDesc>` décrit le document TEI lui-même. Au moins, cette description doit porter sur les trois aspects suivants :

1. le titre et la responsabilité de la ressource numérique (`<titleStmt>`)
2. la distribution de la ressource, y compris les droits d'utilisation (`<publicationStmt>`)
3. le document source dont le texte la ressource numérique représente (`<sourceDesc>`)

Un document TEI peut présenter d'autres éléments dans le `<teiHeader>` afin d'apporter encore plus de détail bibliographique. Par exemple, le `<editionStmt>` précise l'édition de l'œuvre que la ressource représente. Les documents TEI produits par le pipeline *Gallic(orpor)a* ne profitent pas de cet élément parce que certains documents du corpus traité, tel que les manuscrits, n'ont pas d'édition et on veut que chaque document TEI ait les mêmes types de métadonnées dans le `<teiHeader>`. Au contraire, d'autres éléments facultatifs, tel que le `<extent>`, sont produits par le pipeline. L'élément `<extent>` est utile parce qu'il porte sur la taille de la ressource, tel comme le nombre de pages transcrites qu'elle compte. La ressource lexicographique n'est pas toujours une transcription du document complet. Il faut préciser la quantité des pages transcrites.

### 6.1.1 Le titre et la responsabilité (`<titleStmt>`)

Le titre et la responsabilité de la ressource numérique sont tous les deux représentés dans l'élément `<titleStmt>`. Cet élément descendant du `<fileDesc>` peut avoir plusieurs descendants mais il faut au moins un titre. Après le titre (`<title>`) il est recommandé d'indiquer l'individu ou les individus qui sont responsables pour la création du texte représenté et/ou de la ressource numérique. Dans le cadre du projet *Gallic(orpor)a*, nous avons

conçu un schème <titleStmt> simple qui porte sur le titre, l'auteur ou les auteurs du texte, et les personnes du projet responsables pour la création de la ressource numérique.

### La responsabilité

Entre les lignes 4 et 21 de la Figure 7.9 on voit l'élément <respStmt> qui contient des informations sur l'équipe du projet *Gallic(orpor)a*. Tout document encodé par le pipeline s'informe sur le titre (<title>), la personne ou les personnes auxquels est attribuée la responsabilité du texte (<author>), et la responsabilité de notre équipe qui a conçu le pipeline et l'application alto2tei (<respStmt>). La déclaration de responsabilité peut être personnalisée selon le projet ou selon l'équipe qui utilise le pipeline *Gallic(orpor)a* ou l'application alto2tei pour créer la ressource numérique. En général, elle devrait contenir une phrase qui résume la nature de la création de la ressource, telle que la phrase « Transformation from ALTO4 to TEI by », balisée dans l'élément <resp>. Ensuite, elle devrait contenir des éléments <persName> qui renseignent sur les individus responsables, surtout en indiquant leurs noms.

```
1  <titleStmt>
2    <title>Titre du document source traité</title>
3    <author>Auteur</author>
4    <respStmt>
5      <resp>Transformation from ALTO4 to TEI by</resp>
6      <persName>
7        <forename>Kelly</forename>
8        <surname>Christensen</surname>
9        <ptr type="orcid" target="000000027236874X"/>
10     </persName>
11     <persName>
12       <forename>Simon</forename>
13       <surname>Gabay</surname>
14       <ptr type="orcid" target="0000000190944475"/>
15     </persName>
16     <persName>
17       <forename>Ariane</forename>
18       <surname>Pinche</surname>
19       <ptr type="orcid" target="0000000278435050"/>
20     </persName>
21   </respStmt>
22 </titleStmt>
```

FIGURE 6.1 – Les informations sur le titre de la ressource

### Le titre

La ressource produite par le pipeline *Gallic(orpor)a* a besoin d'un titre par lequel elle peut être exploitée. L'application alto2tei que j'ai créée lui attribue le nom du document source dont le texte la ressource représente. Le schème TEI permet de construire

un nouveau titre lors de l'encodage ou d'associer plusieurs titres au document TEI. Cependant, nous avons pris la décision d'emprunter le titre du document source tel qu'il se donne par les sources externes des métadonnées. Au lieu d'aller en détail, dont la TEI se permet, le pipeline simplifie sa manière d'intituler l'encodage en utilisant le même nom associé au document qu'il transcrit.

Dans le schème TEI plusieurs titres peuvent être indiqués dans le `<titleStmt>` afin de détailler plusieurs aspects de l'encodage. Par exemple, le projet *The Bodelian First Folio* a encodé les premières éditions des drames de Shakespeare en TEI et chacun porte plusieurs types de titre.[1] L'encodage de la comédie *Twelfth Night* possède un titre du type "*statement*" qui représente le titre tel qu'il se trouve sur l'imprimé historique (ligne 2, Fig. 6.2). Il donne aussi une variante du titre qui se trouve également dans la source (ligne 2, Fig. 6.2). Enfin, l'encodage présente le titre qui sert à identifier la source aux archives, « Bodelian First Folio, Arch. G c.7 » (ligne 4, Fig. 6.2)

```
1 <titleStmt>
2   <title type="statement">Twelfe Night, or What You Will from Mr. William
      Shakespeares comedies, histories, &amp;amp; tragedies. Published according
      to the true originall copies.</title>
3   <title type="variant">Mr. VVilliam Shakespeares comedies, histories, &amp
      ; tragedies</title>
4   <title type="distinctive">Bodleian First Folio, Arch. G c.7</title>
5 <!-- ... -->
6 </titleStmt>
```

FIGURE 6.2 – Les informations sur le titre d'un imprimé[2]

Pour un manuscrit, l'attribution d'un titre pourrait obliger la création d'un titre qui ne se trouve pas sur le document source. Par exemple, le projet *CatCor* qui a encodé des lettres écrites par Catherine II de la Russie a choisi d'attribuer aux encodages un titre qui s'appuie sur l'identifiant du document source. Dans l'encodage d'une lettre destinée à Fréderick II le 21 juillet 1744, le `<title>` dans le `<titleStmt>` est un titre qui n'apparaît nulle parte sur la source (ligne 2, Fig. 6.3).[3] Contrairement à l'encodage de l'imprimé de Shakespeare, l'encodage de la lettre manuscrite ne donne pas de type au titre. La classification d'un élément `<title>` avec l'attribut `@type` n'est pas nécessaire mais elle est recommandée s'il y a plusieurs titres.

Afin d'encoder les éléments `<title>` à l'échelle, il faut qu'un logicielle (1) ait d'accès aux métadonnées et (2) sache la nature des titres associés au document source. Certains corpus auront d'accès aux métadonnées déjà classifiées. Le catalogue général de la BnF,

1. *The Bodelian First Folio : Digital Facsimile of the First Folio of Shakespeare's Plays*. URL : `http://firstfolio.bodleian.ox.ac.uk/` (visité le 27/08/2022).
2. ibid.
3. CatCor PROJECT. *Letter 02633 : To Frederick II (the Great), 21 July 1744*. Sous la dir. d'Andrew KAHN et RUBIN-DETLEV. 2021. URL : `https://catcor.seh.ox.ac.uk/id/letter-02633`.
4. ibid.

```
<titleStmt>
  <title>CatCor Project: letter-02633</title>
<!-- ... -->
</titleStmt>
```

FIGURE 6.3 – Les informations sur le titre d'un manuscrit [4]

par exemple, organise ses métadonnées dans une structure de données XML UNIMARC. Chaque titre associé au document est balisé dans des éléments XML qui portent sur le type du titre. Par exemple, l'UNIMARC présente le type « titre uniforme » dans l'élément `<mxc:datafield tag="500">` et le type « titre de forme » dans l'élément `<mxc:datafield tag="503">`. En récupérant les métadonnées depuis une source ainsi organisée, un logiciel pourrait attribuer un type à l'élément `<title>`. Cependant, ce qui est le cas pour le corpus du projet *Gallic(orpora)*, si certains documents traités n'ont pas de métadonnées accessibles dans un tel format, un logiciel ne peut pas parvenir au tel détail dans le `<titleStmt>`.

### L'auteur

Après le titre, le `<titleStmt>` renseigne sur l'individu ou les individus aux lesquels la propriété intellectuelle du document source est attribuée. Cette donnée est encodée dans l'élément `<author>`. L'encodage peut être si minimal que l'élément `<author>` ne contient que du texte ou de l'élément simple `<name>` qui ensuite contiendrait du texte non annoté. (cf. Figure 6.4) Sinon d'autres éléments TEI peuvent baliser et apporter plus de détail sur les composants du nom de l'auteur. (cf. Figure 6.5)

```
<author>
  <name>Donatien Alphonse François de Sade</name>
</author>
```

FIGURE 6.4 – L'auteur simple

```
<author xmlid="Sa1">
  <persName>
    <forename>Donatien Alphonse François</forename>
    <nameLink>de</nameLink>
    <surname>Sade</surname>
    <ptr type="isni" target="0000000084961458"/>
  </persName>
</author>
```

FIGURE 6.5 – L'auteur enrichi

Un élément qui descend souvent du `<author>` est le `<ptr>` qui veut dire *pointer* en anglais. Il indique une ressource ou une donnée externe, tel que l'identifiant ISNI, afin d'enrichir

les informations de l'objet auquel il est attaché. On voit un exemple du *pointer* sur les lignes 9, 14, et 19 de la Figure 7.9 où l'élément indique l'ORCID unique de l'individu responsable pour la création de la ressource numérique.

La Figure 6.5 montre l'exemple d'un document réel TEI produit par le pipeline *Gallic(orpor)a*. Le nom de l'auteur est divisé en trois composants, selon les données UNIMARC fournies par le catalogue général de la BnF. Le catalogue désigne *Donatien Alphonse François de* comme la « partie du nom autre que l'élément d'entrée ». L'UNIMARC balise cette partie secondaire dans l'élément `<mxc:subfield code="b">` de l'élément `<mxc:datafield tag="700">` ou `<mxc:datafield tag="701">`. [5] Mon application `alto2tei`, que j'ai développée pour le pipeline *Gallic(orpor)a*, a ensuite tiré la partie *de* du nom puisqu'elle est un lien entre ses composants et peut donc être encodé dans l'élément TEI `<nameLink>`. Le catalogue général a reconnu le nom *Sade* comme le nom d'entrée de l'auteur, que l'UNIMARC balise dans l'élément `<mxc:subfield code="a">`. Le document TEI du pipeline *Gallic(orpor)a* a donc balisé cette donnée dans l'élément `<surname>`, c'est-à-dire le nom de famille.

Un défi de l'application `alto2tei` était la présentation des données portant sur les auteurs, surtout quand il y en avait plusieurs. Dans le schème TEI, le `<titleStmt>` peut renseigner sur plusieurs auteurs en répétant l'élément comme dans la Figure 6.6. Le schème TEI permet d'indiquer le rôle de chaque auteur listé dans un `<titleStmt>`. Mais, défaut de métadonnées, l'application `alto2tei` et le pipeline *Gallic(orpor)a* ne s'appuient pas sur cette donnée. Même s'il serait intéressant, toute autrice nommée et tout auteur nommé dans le `<teiHeader>` n'a pas de relation détaillée au document source.

Il y a deux raisons pour ainsi modéliser les données sur les auteurs. Dans un premier temps, l'application `alto2tei` recherche la catégorie d'auteur dans les données renvoyées par l'API IIIF dans ce qui s'appelle le *manifest*. Renvoyées en format JSON, les données du *manifest* sont souvent très minimales, n'ayant que le nom de l'autrice ou de l'auteur. De temps en temps les données du *manifest* IIIF ont des dates de l'individu. Mais le seul aspect sur lequel on peut compter est le nom. Après l'API IIIF, l'application recherche les données dans le catalogue général de la BnF en utilisant son API SRU. Les données du catalogue sont bien plus détaillées, mais le schème UNIMARC ne permet pas d'enregistrer de l'égalité entre plusieurs auteurs ni préciser la nature de leur contribution au document. Dans l'exemple de la Figure 6.6, les données UNIMARC de la BnF n'indiqueraient pas que Giacomo Meyerbeer est le compositeur de l'opéra *Les Huguenots* ni que lui et le librettiste Eugène Scribe partagent en parts égales la responsabilité pour l'œuvre.

5. *Manuel UNIMARC : format bibliographique*. Transition bibliographique - Programme national. URL : `https://www.transition-bibliographique.fr/unimarc/manuel-unimarc-format-bibliographique/` (visité le 27/08/2022).

```
<titleStmt>
  <title>Les Huguenots</title>
  <author xml:id="Me1">
    <persName>
      <forename>Giacomo</forename>
      <surname>Meyerbeer</surname>
      <ptr type="isni" target="0000000122817116"/>
    </persName>
  </author>
  <author xml:id="Sc1">
    <persName>
      <forename>Eugène</forename>
      <surname>Scribe</surname>
      <ptr type="isni" target="000000012122970X"/>
    </persName>
  </author>
  <author xml:id="De1">
    <persName>
      <forename>Émile</forename>
      <surname>Deschamps</surname>
      <ptr type="isni" target="0000000122807567"/>
    </persName>
  </author>
  <author xml:id="Ro1">
    <persName>
      <forename>Gaetano</forename>
      <surname>Rossi</surname>
      <ptr type="isni" target="0000000121219499"/>
    </persName>
  </author>
<!-- ... -->
</titleStmt>
```

FIGURE 6.6 – Plusieurs auteurs dans un `<titleStmt>`

### 6.1.2 La taille de la ressource numérique (`<extent>`)

Lors de la reconnaissance du texte, les modèles HTR génèrent un certain nombre de fichier ALTO. Grâce à l'unité indiqué comme « images » dans l'élément `<measure>`, la taille est calculée facilement à partir du compte de fichiers ALTO produits et ensuite traités pour que leurs données soient contenues dans l'élément `<sourceDoc>` de la ressource numérique.

```
<extent>
  <measure unit="images" n="20"/>
</extent>
```

FIGURE 6.7 – La taille de la ressource

### 6.1.3 La distribution de la ressource numérique (<publicationStmt>)

L'élément <publicationStmt> contient des données importantes qui renseignent sur la distribution et les droits d'utilisation de la ressource numérique. Toute métadonnée contenue dedans porte sur le contexte de la création de la ressource, aucune sur le document source représenté. Pour toute ressource produite par le pipeline, les données du <publicationStmt> ne doivent pas changer si le projet qui a démarré le pipeline. L'exception est la date de la création de la ressource, que l'application alto2tei génère automatiquement.

Comme montre la Figure 6.8, trois données du <publicationStmt> peuvent être personnalisées. L'entité reconnu comme l'éditeur (*publisher*) de la ressource peut être changé selon le projet. Dans l'exemple de la Figure 6.8, le *publisher* est « Gallic(orpor)a ». L'autorité qui l'a financé et qui est civilement responsable pour la ressource est le Data-Lab de la BnF, que l'élément <authority> indique. Enfin, les droits d'utilisation de la ressource sont indiquées par les éléments <availability> et <licence>.

```
<publicationStmt>
  <publisher>Gallic(orpor)a</publisher>
  <authority>BnF DATALab</authority>
  <availability status="restricted" n="cc-by">
    <licence target="https://creativecommons.org/licenses/by/4.0/"/>
  </availability>
  <date when="2022-07-29"/>
</publicationStmt>
```

FIGURE 6.8 – Plusieurs auteurs dans un <publicationStmt>

### 6.1.4 Le document source (<sourceDesc>)

Le dernier aspect de la description bibliographique du <teiHeader> porte sur le document source dont le texte est représenté. Cet aspect se balise dans l'élément <sourceDesc> qui est le dernier élément du <fileDesc> que l'application alto2tei génère. Cet élément compte sur le plus grand nombre de sources externes afin d'informer une diversité des métadonnées. La description bibliographique de la source porte sur sa création et sur sa conservation actuelle.

#### La citation bibliographique (<bibl>)

Dans un premier temps, la description de la source s'appuie sur les aspects de sa création. En répétant certaines informations du <titleStmt>, l'élément <bibl> présente (1) les individus auxquels est attribuée la propriété intellectuelle du document, (2) le titre du document source, (3) son lieu de publication, (4) l'éditeur, et (5) la date de publication. L'arborescence du <bibl> est montrée dans la Figure 6.9.

Puisque la notice du catalogue pour le document source montré dans l'exemple de la Figure 6.9 peut être trouvée par l'application `alto2tei`, les données du `<bibl>` sont plus complète qu'elle auraient été si l'application avait besoin de compter uniquement sur le *manifest* IIIF. La date de publication ou d'apparition, par exemple, représentée dans l'élément `<date>`, porte des attributs bien détaillés. Les données UNIMARC du catalogue général de la BnF indiquent la certitude qu'a la bibliothèque quant à la date déclarée. Au lieu d'extraire uniquement la date des données UNIMARC, l'application `alto2tei` traite ces autres données et détermine la certitude de la date, « low », « medium », ou « high ».

L'exemple de la Figure 6.9 est d'un imprimé du XVIIIe siècle. Il a donc été publié par un éditeur. L'éditeur, par contre, est indiqué dans les données UNIMARC de la BnF comme « s.n. » qui indique que la bibliothèque n'est pas certain de l'éditeur. L'application `alto2tei` ne peut pas traiter les données dont elle ne se dispose pas ni veut elle transformer la donnée « s.n. » dans un autre format au cas où elle corrompt la signification. Par conséquent, l'élément `<publisher>` contient la donnée du catalogue même si elle indique une manque d'information.

Le `<bibl>` d'un manuscrit contient les mêmes éléments que celui d'un imprimé. Même si un manuscrit n'a pas d'éditeur ni n'est pas publié de la même manière qu'un imprimé, sa citation bibliographique porte les mêmes éléments `<pubPlace>` et `<publisher>`. Normalement, le catalogue général de la BnF n'indiquera pas de donnée pour ces aspects. Quand la donnée n'est pas disponible, l'application `alto2tei` garde toujours l'arborescence généralisée de la ressource numérique, mais elle indique dans l'élément que la donnée n'était pas trouvée.

### La description de la source physique (`<msDesc>`)

Dernièrement, le document source physique et son fac-similé numérique sont tous les deux indiqués dans l'élément `<msDesc>`. Tandis que la citation bibliographique (`<bibl>`) porte sur le document en tant qu'une œuvre littéraire créée, la *description du manuscrit* (`<msDesc>`) porte sur le document en tant qu'un objet réel dans le monde. Il a deux enfants principaux qui sont montrés dans la Figure 6.10. Premièrement, l'élément `<msIdentifier>` sert à identifier le document physique ou le fac-similé numérique dans un catalogue quelque part. Deuxièmement, l'élément `<physDesc>` sert à décrire le type du document, soit manuscrit soit imprimé.

L'identification du document traité et encodé est très importante. Par exemple, un imprimé aura plusieurs exemplaires ; chacun pourrait avoir des différences et produirait une transcription distincte. Afin de bien indiquer quel objet de texte a été traité par le pipeline *Gallic(orpor)a*, il faut identifier le document source physique dont le fac-similé numérique les modèles HTR ont saisi. Les éléments `idno` dans le `<msDesc>` indiquent l'identifiant d'un document. L'identifiant du document source physique est l'élément principal descendant du `<msIdentifier>`. L'identifiant du fac-similé numérique de la base de

```
<sourceDesc>
  <bibl>
    <ptr target="http://catalogue.bnf.fr/ark:/12148/cb30369299r"/>
    <author ref="#Re1">
      <persName>
        <forename>Jean-François</forename>
        <surname>Regnard</surname>
        <ptr type="isni" target="000000012118509X"/>
      </persName>
    </author>
    <author ref="#Du2">
      <persName>
        <forename>Charles</forename>
        <surname>Du Fresny</surname>
        <ptr type="isni" target="0000000140935001"/>
      </persName>
    </author>
    <title>Scènes françoises de la comédie italienne intitulée "la Foire S
    .-Germain" , comme elles ont paru dans les premières représentations</
    title>
    <pubPlace key="FR">Grenoble</pubPlace>
    <publisher>[s.n.]</publisher>
    <date when="1696" cert="high" resp="BNF">1696</date>
  </bibl>
  <msDesc>
<!-- ... -->
  </msDesc>
</sourceDesc>
```

FIGURE 6.9 – La citation bibliographique (`<bibl>`)

données Gallica est indiqué dans le <altIdentifier> puisque le fac-similé est le document dérivé du document source et donc n'est pas le document indiqué par le <idno> principal. Le <idno> alternatif est l'ARK du fac-similé numérique sur Gallica. Le <idno> principal du <msDesc> est le cote du document physique selon le catalogue de la Bibliothèque nationale de France.

```
<sourceDesc>
  <bibl>
<!-- ... -->
  </bibl>
  <msDesc>
    <msIdentifier>
      <country key="FR"/>
      <settlement>Paris</settlement>
      <repository>Bibliothèque nationale de France</repository>
      <idno>YF-5877</idno>
      <altIdentifier>
        <idno type="ark">bpt6k1281160s</idno>
      </altIdentifier>
    </msIdentifier>
    <physDesc>
      <objectDesc>
        <p>Texte imprimé</p>
      </objectDesc>
    </physDesc>
  </msDesc>
</sourceDesc>
```

FIGURE 6.10 – La description de la source (<msDesc>)

## 6.2 La description non bibliographique (<profileDesc>)

Après la description du fichier (<fileDesc>) qui porte sur les détails bibliographiques de la ressource numérique, les métadonnées du <teiHeader> renseignent sur une description non bibliographique de la ressource. Normalement la description non bibliographique s'informe des langues utilisées dans le texte représenté. Elle peut aussi s'informer des lieux ou personnages référencés dans le texte. Mais ces détails exigent une analyse linguistique, sinon littéraire, du texte. Un jour l'analyse linguistique du pipeline, en particulier la reconnaissance des entités nommées, sera si perfectionnée et si fiable que le <profileDesc> de la ressource numérique contiendra les listes de lieux et de noms dans le texte. Actuellement, le <profileDesc> renseigne simplement sur la langue du texte, et l'application alto2tei que j'ai créée ne peut porter que sur une seule langue. Il est normal que les ingénieurs qui reprennent ce travail après mon stage aillent évoluer notre modélisation actuelle du <profileDesc> pour qu'elle ait des données portant sur plusieurs langues du texte ainsi que sur les noms propres et les lieux nommés dans le texte.

La Figure 6.12 montre l'exemple d'un `<profileDesc>` complété par l'application `alto2tei` dans le cadre du projet *Gallic(orpor)a*. En prenant le même document dont les autres parties de l'encodage se voient dans les Figures 6.9 et 6.10, la langue identifiée est français. Cette langue a été récupérée du *manifest* IIIF du fac-similé numérique sur Gallica. Mais quand l'application `alto2tei` peut aussi accéder aux données du catalogue général de la BnF, l'identifiant de la langue est souvent disponible dans les données UNIMARC. L'identifiant « fre » indique le français moderne. Même si la langue identifiée dans le *manifest* IIIF était encore français, les données UNIMARC du catalogue de la BnF pourrait préciser qu'il est du moyen français, qui est représenté par l'identifiant « frm ».

```
<profileDesc>
  <langUsage>
    <language ident="fre">français</language>
  </langUsage>
</profileDesc>
```

FIGURE 6.11 – La description non bibliographique de la source (`<profileDesc>`)

## 6.3 La description technique (`<encodingDesc>`)

Le dernier composant du `<teiHeader>` est une description technique de l'encodage. Balisée dans l'élément `<encodingDesc>`, la description technique informe de la manière par laquelle l'encodage a été produit. Tout projet scientifique devrait pouvoir reproduire ses résultats. La ressource doit donc attester à la manière par laquelle les résultats de ses prédictions HTR a été faits. L'élément TEI `<appInfo>` renseigne sur le logiciel HTR qui prédit du segment et du texte sur les images numériques. De l'information sur les images et leur origine sont décrites dans les autres éléments du `<teiHeader>`. L'information sur l'appareil qui les a traité est contenu dans le `<appInfo>`.

En plus du logiciel, le `<encodingDesc>` informe du syntaxe par lequel le texte a été encodé en utilisant l'élément `<classDecl>`. La *déclaration des classes* (`<classDecl>`) porte sur les classes visées à structurer et organiser le texte prédit. Dans le cadre du projet *Gallic(orpor)a*, les classes du vocabulaire *SegmOnto* sont attribuées aux lignes de texte et aux zones dans lesquels le texte s'est trouvé. Puisque les lignes de texte et les zones portent les noms du vocabulaire *SegmOnto*, le `<classDecl>` explique fournit une description de chaque classe. La Figure 6.12 montre comment la description de classe est balisé dans l'élément `<catDesc>` qui lui-même se trouve balisé dans la catégorie (`<category>`) de la classe, soit la ligne soit la zone.

```
<encodingDesc>
  <appInfo>
    <application ident="Kraken" version="3.0.13">
      <label>Kraken</label>
      <ptr target="https://github.com/mittagessen/kraken"/>
    </application>
  </appInfo>
  <classDecl>
    <taxonomy xml:id="SegmOnto">
      <bibl>
        <title>SegmOnto</title>
        <ptr target="https://github.com/segmonto"/>
      </bibl>
      <category xml:id="SegmOntoZones">
        <catDesc xml:id="MainZone">
          <title>MainZone</title>
          <ptr target="https://segmonto.github.io/gd/gdZ/MainZone"/>
        </catDesc>
        <catDesc xml:id="TitlePageZone">
          <title>TitlePageZone</title>
          <ptr target="https://segmonto.github.io/gd/gdZ/TitlePageZone"/>
        </catDesc>
<!-- more SegmOnto Zones -->
      </category>
      <category xml:id="SegmOntoLines">
        <catDesc xml:id="DefaultLine">
          <title>DefaultLine</title>
          <ptr target="https://segmonto.github.io/gd/gdL/DefaultLine"/>
        </catDesc>
        <catDesc xml:id="HeadingLine">
          <title>HeadingLine</title>
          <ptr target="https://segmonto.github.io/gd/gdL/HeadingLine"/>
        </catDesc>
<!-- more SegmOnto Lines -->
      </category>
    </taxonomy>
  </classDecl>
</encodingDesc>
```

FIGURE 6.12 – La description non bibliographique de la source (<profileDesc>)

# Troisième partie

# Mise en opérationnelle du projet

# Chapitre 7

# La génération du `<teiHeader>`

J'ai créé l'application `alto2tei` afin de compléter le pipeline du projet *Gallic(orpor)a*. Pour résumer, la première étape du pipeline est la récupération des pages numérisées du document source. Ensuite il prédit le texte et la mise en page du fac-similé numérique en traitant chaque page avec les modèles HTR. Les données produites par les modèles sont en format XML ALTO. Mais le pipeline veut présenter les données enrichies par les métadonnées et par l'analyse linguistique en format XML TEI puisque le schème TEI est plus utilisé et il convient mieux à l'édition du texte que le schème ALTO. Mon application `alto2tei` a complété le pipeline en construisant un document TEI à partir des données des modèles HTR.

Cependant, les données du text transcrit ne constituent pas tout ce qu'il faut pour construire un document TEI. Il faut aussi des métadonnées qui s'encadrent dans l'élément TEI `<teiHeader>`. L'application `alto2tei` a donc besoin de récupérer les métadonnées à partir des sources externes, qui contiennent des données autre que celles créées par le pipeline. Pour y parvenir, l'application s'appuie sur la technologie d'API (Application Programming Interface) qui lui permet de poser des questions aux sources en ligne et de comprendre leur réponse.

Quelles métadonnées faut-il chercher ? Dans le chapitre 6, je parle de trois objets de texte conceptuels pertinents : (1) la ressource numérique elle-même en format XML TEI, (2) le fac-similé numérique disponible depuis l'API de la BnF dont les images sont traités par les modèles HTR, et (3) le document physique qui est conservé par la BnF et qui a été numérisé. Chacun est distinct et ses métadonnées se trouvent depuis des sources différentes. Afin d'informer des métadonnées de la ressource numérique, il faut donc s'appuyer sur les métadonnées de ses trois objets de texte.

Le pipeline du projet *Gallic(orpor)a* a besoin de diversifier sa manière de récupérer des métadonnées. L'application `alto2tei` profite de quatre sources externes de données afin de renseigner sur les trois documents conceptuels concernés.

1. les métadonnées sur la ressource lexicographique numérique
   — source : fichier de configuration personnalisable

   - format : YAML (*Yet Another Markup Language*)
   - portée : informations administratives portant sur la création de la ressource, y compris les droits d'utilisation et les autorités qui s'en chargent
2. les métadonnées sur le fac-similé numérique
   - source : manifest IIIF renvoyé de l'IIIF API
   - format : JSON (*JavaScript Object Notation*)
   - portée : données bibliographiques qui servent à identifier les exemplaires numériques traités par des modèles HTR
3. les métadonnées sur le document source physique
   - source : réponse à la requête envoyée à l'API SRU (*Search/Retrieve via URL*) de la BnF, qui interroge le catalogue général de la BnF
   - format : UNIMARC XML
   - portée : données bibliographiques, y compris la responsabilité du document source, sa création, et sa conservation actuelle

Les métadonnées sur le document source physique s'appuient encore sur une autre source de données afin de récupérer où se trouve l'institution qui héberge le document. Dans le cadre du projet *Gallic(orpor)a*, la réponse à cette question est toujours Paris, puisque la BnF se situe au capital. Mais, afin d'éviter l'encodage dur, l'application `alto2tei` recherche cette donnée dans la base de données du Système Universitaire de Documentation (SUDOC) puisque la localisation de l'institution hôte du document physique n'est pas encodée dans les données UNIMARC selon l'usage de la BnF.

## 7.1 La récupération des métadonnées

Comme s'explique dans le chapitre 4, l'application `alto2tei` s'appuie sur un système de fichier fixé. Cela lui permet d'accéder à l'identifiant ARK (Archival Resource Key) du document dont les pages numérisées ont été traitées par les modèles HTR. Cet identifiant donne la clef aux données sur le fac-similé numérique encodées dans le *manifest* IIIF (International Image Interoperability Framework). L'une des données dans le *manifest* IIIF, au moins pour les documents de la base de données Gallica, porte sur la relation entre le fac-similé numérique est la source physique à partir duquel le fac-similé numérique a été produit. (cf. Fig. 7.1) Cette relation permet de rechercher les métadonnées du document physique dans le catalogue général de la BnF.

Si la relation entre le document numérique sur Gallica et le document physique dans l'un des magasin des la BnF n'est pas établie, l'application `alto2tei` n'abandonne pas tout essai de produire un document TEI. Au lieu de s'arrêter, elle compte uniquement sur les données du *manifest* IIIF. Ces données sont vérifiées puisqu'elles portent sur le fac-similé numérique dont les images les modèles HTR du pipeline ont traité. Le fac-similé numérique est toujours l'un des objets de texte pertinents aux métadonnées de la

```
"metadata": [
  {
  "label": "Relation",
  "value": "Notice du catalogue : http://catalogue.bnf.fr/ark:/12148/
    cb30369299r"
  },
]
```

FIGURE 7.1 – Une partie du *manifest* IIIF pour le fac-similé numérique d'ARK `bpt6k1281160s`, un imprimé numérisé sur Gallica

ressource. L'application `alto2tei` peut donc créer deux genres de `<teiHeader>`, l'un avec un maximum d'information et l'autre, au cas où la relation entre le fac-similé numérique et le document physique n'était pas établie, avec un peu moins d'information.

En tout cas, chaque genre de `<teiHeader>` aura la même arborescence parce que tout élément est d'abord créé avec des données par défaut, telle que la phrase « *Information not available* ». La donnée par défaut est ensuite remplacée par la donnée récupérée des sources externes. Par conséquent, l'application `alto2tei` ne s'arrête pas si une donnée n'est pas récupérée. Il peut arriver, par exemple, que la relation entre le document physique et le fac-similé numérique est établie et les données du catalogue sont traitées, mais une donnée en particulière n'est pas disponible dans la notice du document physique. Dans ce cas, le `<teiHeader>` aura des éléments qui contiennent une phrase par défaut même si des autres éléments contiennent des données récupérées depuis la source ciblée. L'application `alto2tei` essaie de fournir autant des métadonnées que possible et surmonter les particularités de la disponibilité des données.

## 7.2 Les sources des métadonnées

Quatre source de données servent à informer des éléments du `<teiHeader>`. Elles sont le *manifest* IIIF, les données UNIMARC du catalogue général de la BnF, la base de données du site web du Système Universitaire de Documentation (SUDOC), le fichier de configuration, et les fichiers ALTO produits par des modèles HTR. Les trois premières sources exigent une requête au bon API. Le *manifest* IIIF est renvoyé par une requête à l'API IIIF du fac-similé numérique sur Gallica, selon les normes imposées par la BnF. Les données UNIMARC du catalogue général de la BnF sont accessible à partir d'une requête à l'API SRU (*Search/Retrieve via URL* de la BnF. Enfin, les données du Système Universitaire de Documentation sont renvoyées à partir d'une requête à l'*endpoint* API du site SUDOC. L'application `alto2tei` se dispose directement du fichier de configuration et des fichiers ALTO qu'elle traite.

### 7.2.1 Le format du *manifest* IIIF

Le *manifest* IIIF est un format de données standardisés qui est envoyé par tout API IIIF selon le syntaxe de requête demandé par l'organisation qui gère l'API. La Bibliothèque nationale de France exige le syntaxe suivant pour toute requête envoyée à l'API IIIF des fac-similés numériques sur Gallica :

`https://gallica.bnf.fr/iiif/ark:/12148/<ARK>/manifest.json`

Pour utiliser cette requête, il faut simplement remplacer la partie `<ARK>` par l'ARK du document cherché. Grâce au système de fichier du pipeline *Gallic(orpor)a*, l'ARK du document est facilement disponible. Il est le nom du dossier qui contient les fichiers ALTO traités.

La structure du *manifest* IIIF peut varier un peu, mais certains composants sont toujours attendus. Le Tableau 7.2 montre toute clef principale du *manfiest* pour un fac-similé numérique mis en ligne sur Gallica par la BnF. L'exemple prend le livret imprimé du ballet *Les divers entretiens de la fontaine de Vaucluse* de 1649, dont le fac-similé a l'identifiant `bpt6k1513919s`. Les parties les plus importants sont aussi les plus complexes, la clef *metadata* et la clef *sequences*. La première contient une suite des pairs clef-valeur, chacun porte sur une métadonnée du fac-simlié numérique. La dernière s'appuie sur les images numériques. Pour l'application `alto2tei` et la création du `<teiHeader>` les métadonnées du *manifest* sont l'essentiel.

| **clef** | **valeur** |
|---|---|
| @id | https ://gallica.bnf.fr/iiif/ark :/12148/bpt6k1513919s/manifest.json |
| label | BnF, département Réserve des livres rares, RES-YF-1990 |
| attribution | Bibliothèque nationale de France |
| license | https ://gallica.bnf.fr/html/und/conditions-dutilisation-des-contenus-de-gallica |
| logo | https ://gallica.bnf.fr/mbImage/logos/logo-bnf.png |
| related | https ://gallica.bnf.fr/ark :/12148/bpt6k1513919s |
| description | Les divers entretiens de la fontaine de Vaucluse : ballet dansé à la grande salle du Roure l'année 1649... |
| metadata | *suite des pairs qui portent sur les métadonnées du fac-similé numérique* |
| sequences | *imbrication complexe pour contenir des données sur les images, dit canvases, du fac-similé numérique* |
| thumbnail | *pair qui contient un lien pour l'aperçu du document* |
| context | http ://iiif.io/api/presentation/2/context.json |

FIGURE 7.2 – Les clefs principales du *manifest* IIIF d'un document sur Gallica (ARK `bpt6k1513919s`)

Le contenu du *metadata* du *manifest* IIIF peut varier de document à document, même quand les données sont encodées par la même institution telle que la Bibliothèque nationale de France. Comme s'explique dans le chapitre 6, il faut déterminer l'essentiel

quant aux métadonnées à mettre dans le schème TEI. Il faut chercher les métadonnées qui s'appliquent tous les deux aux manuscrits et aux imprimés. En plus de cela, il faut aussi chercher les métadonnées qui, selon l'usage de la BnF, sont normalement encodées dans le *manifest* IIIF du document sur Gallica. J'ai déterminé que l'essentiel du *manifest* IIIF sont les métadonnées listée dans le Tableau 7.3.

| ***label*** | ***value*** |
|---|---|
| Relation | Notice du catalogue : http ://catalogue.bnf.fr/ark :/12148/cb33352789b |
| Repository | Bibliothèque nationale de France |
| Shelfmark | Bibliothèque nationale de France, département Réserve des livres rares, RES-YF-1990 |
| Title | Les divers entretiens de la fontaine de Vaucluse : ballet dansé à la grande salle du Roure l'année 1649 |
| Language | français |
| Creator | Nouguier, de. Auteur du texte |
| Date | 1649 |

FIGURE 7.3 – Les métadonnées essentielles du *manifest* IIIF (ARK `bpt6k1513919s`)

Certaines métadonnées du *manifest* posent des avantages et des défis. L'un des avantages est la manière par laquelle la BnF encode la donnée *Relation* pour les documents mis sur Gallica. La donnée *Relation* peut se répéter parce que le fac-similé peut être lié aux plusieurs notices du catalogue. Cependant, elle termine toujours par l'ARK unique du document physique auquel le fac-similé est lié. Ainsi, l'application `alto2tei` se dispose de la donnée *Relation* et, en traitant la fin de la chaîne, extrait l'ARK du document physique dans le catalogue général de la BnF. Un défi à surmonter est quand un type (*label*) de métadonnée n'est pas répété mais il contient plusieurs valeurs. Cela peut arriver quand un document a plusieurs auteurs. Pour cette raison, l'application `alto2tei` traite l'encodage de toute donnée sur l'auteur d'une manière différente que les autres métadonnées.

### 7.2.2 Le format UNIMARC

Étant communiquées en XML, les données du catalogue général de la BnF se trouvent sur l'arbre de la notice en format UNIMARC ; ce format leur donne le préfix `mxc` qui fait référence au *MarcXchange*. Un exemple des données UNIMARC renvoyées par l'API SRU de la BnF est montré dans la Figure 7.4b. L'exemple prend toujours le livret imprimé du ballet *Les divers entretiens de la fontaine de Vaucluse* de 1649. Le fac-similé numérique sur Gallica a l'identifiant `bpt6k1513919s`. Mais, comme expliqué ci-dessus, l'identifiant du document physique est récupéré en examinant le *manifest* IIIF du fac-similé numérique. L'ARK du document physique (`cb33352789b` dans l'exemple) permet d'interroger les données du catalogue général de la BnF grâce à son API *Search/-Retrieve via URL* (SRU).

```python
class SRU:
  def __init__(self, ark):
    """Args:
      ark (string): document ARK in BnF catalogue"""
    self.ark = ark

  def request(self):
    r = requests.get(f'http://catalogue.bnf.fr/api/SRU?version=1.2&
    operation=searchRetrieve&query=(bib.persistentid all "ark:/12148/{self.
    ark}")')
```

(a) La requête envoyée à l'API SRU par l'application python `alto2tei`

```xml
<srw:searchRetrieveResponse xmlns:srw="http://www.loc.gov/zing/srw/" xmlns=
    "http://catalogue.bnf.fr/namespaces/InterXMarc">
<!-- ... -->
  <srw:echoedSearchRetrieveRequest>
    <srw:query>(bib.persistentid all "ark:/12148/cb33352789b")</srw:query>
  </srw:echoedSearchRetrieveRequest>
  <srw:numberOfRecords>1</srw:numberOfRecords>
  <srw:records>
    <srw:record>
      <srw:recordSchema>marcxchange</srw:recordSchema>
      <srw:recordPacking>xml</srw:recordPacking>
      <srw:recordData>
        <mxc:record format="Unimarc" id="ark:/12148/cb33352789b" type="
    Bibliographic" xmlns:mxc="info:lc/xmlns/marcxchange-v2">
<!-- data in Unimarc formatting, with prefix "mxc" -->
        </mxc:record>
      </srw:record>
    </srw:recordData>
<!-- ... -->
```

(b) La réponse de l'API SRU pour le document d'ARK `bpt6k1513919s`

FIGURE 7.4 – La requête et la réponse à l'API SRU [version 1.2]

Les données UNIMARC sont emballées dans les éléments du préfix `srw` qui fait référence au protocole SRU. [1] L'élément `<srw:recordData>` (ligne 11, Fig 7.4b) présente les données da la notice du catalogue trouvée. L'application `alto2tei` détermine qu'elle a bien localisé le bon document physique dans le catalogue de la BnF si la réponse de l'API SRU ne contient qu'une seule notice. Le nombre de notices trouvées est sur la ligne 6 de la Figure 7.4b. Si ce nombre est plus que « 1 » cela indique que l'ARK récupéré de la donnée *Relation* du *manifest* IIIF est lié aux notices de plusieurs documents. Ne pouvant pas déterminer quel document est le bon, l'application `alto2tei` abandonne les données UNIMARC et compte uniquement sur les métadonnées du *manifest* IIIF.

1. *SRU : Search/Retrieval via URL – SRU, CQL and ZeeRex (Standards, Library of Congress).* URL : `https://www.loc.gov/standards/sru/` (visité le 02/09/2022).

### 7.2.3 Le format des données du site SUDOC

Puisque les données UNIMARC ne renseignent pas sur le lieu de l'institution qui conserve le document physique, il faut d'autre source de données pour remplir l'élément TEI `<settlement>` du `<sourceDesc>`. Pour trouver une solution, j'en ai discuté avec une spécialiste à la Bibliothèque nationale de France, Florence Tfibel. Dans un courriel, Tfibel a dit que « la localisation de l'institution n'y est effectivement nulle part renseignée "en dur" » dans les données UNIMARC du catalogue.[2] Alors elle a eu l'idée de la chercher sur le site web du Système Universitaire de Documentation qui gère un répertoire des centres de ressources (RCR), auquel appartient la BnF. La BnF se dispose d'un numéro RCR. Son emplacement sur l'arbre des données UNIMARC du catalogue est montré dans la Figure 7.5. Le numéro RCR de la BnF est 759999999.

```
<mxc:datafield tag="930" ind1=" " ind2=" ">
<!-- ... -->
  <mxc:subfield code="b">759999999</mxc:subfield>
<!-- ... -->
</mxc:datafield>
```

FIGURE 7.5 – L'emplacement du RCR sur l'arbre des données UNIMARC

Le site web du Système Universitaire de Documentation ne se dispose pas d'un genre d'API SRU aussi efficace que celui de la BnF, mais les données de son site peuvent être interrogées en traitant la page du centre de ressource qui porte le numéro cherché. Il est possible de lire la page d'un centre en construisant l'URL que le site attend à la suite d'une recherche. Si on recherche le numéro de la BnF, l'URL sera le suivant, ou le dernier chiffre après la signe d'égalité est le numéro RCR de la BnF :

```
http://www.sudoc.abes.fr/cbs/xslt//DB=2.2/
    CMD?ACT=SRCHA&IKT=8888&SRT=RLV&TRM=759999999
```

L'application `alto2tei` demande à l'API du site SUDOC la page de cet URL ; ensuite, elle traite les données HTML renvoyées. (cf. Fig 7.6) La page HTML a un tableau avec une une cellule (`<td>`) qui contient la phrase « Adresse postale » dans le `<span>` d'un `<div>`. À la file après cette cellule il y a une autre cellule avec les données portant sur l'adresse du centre. Ayant la localiser avec l'aide de la première cellule, l'application `alto2tei` traite les données de cette dernière et extrait la ville et le nom uniforme du centre. Même si la ville et le nom de la BnF seront toujours Paris et la Bibliothèque nationale de France, respectivement, il est meilleur de s'appuyer sur les sources de données pour que l'application puisse s'adapter plus facilement aux autres utilisations.

2. Florence TFIBEL. *RE : Tr : [Gallic(Orpor)a] Question Sur l'Unimarc Du Catalogue Généra.* E-mail. 13 juill. 2022.

```
<tr>
  <td class="rec_lable">
    <div>
      <span>Adresse postale : </span>
    </div>
  </td>
  <td clas="rec_title">
    <div>
      <span>Bibliothèque nationale de France</span>
    </div>
    <div>
      <span>quai Francois Mauriac </span>
    </div>
    <div>
      <span>75706 Paris CEDEX 13</span>
    </div>
    <div>
      <span>France</span>
    </div>
    <div>
      <span> </span>
    </div>
  </td>
</tr>
```

FIGURE 7.6 – Les données cherchées depuis le site SUDOC

### 7.2.4 Le format du fichier de configuration

Une partie importante des métadonnées s'appuient aussi sur le fichier de configuration que les utilisateurs de l'application `alto2tei` peut facilement personnaliser. Son format est le YAML (Yet Another Markup Language) qui n'exige pas de grande expertise dans la programmation. Comme l'eXtensible Markup Language (XML) le YAML est ce qu'on appelle un langage de « Mark Up » et donc il imbrique des données. Mais au lieu d'utiliser les éléments XML (`< >`), le YAML utilise simplement les deux points et de la tabulation. Ainsi les utilisateurs de divers spécialités peuvent adapter l'application `alto2tei` et le pipeline *Gallic(orpor)a* aux données de leur propre projet.

Le fichier de configuration informe trois aspects de la création du document TEI. Dans un premier temps, il informe l'application `alto2tei` où il peut trouver les fichiers ALTO qui a été traités par les modèles HTR. Cette donnée est personnalisable pour qu'une utilisatrice ou un utilisateur puisse transformer les transcriptions ALTO stockés sous divers dossiers. Dans un deuxième temps, le fichier de configuration informe la requête que l'application `alto2tei` envoie à l'API IIIF, le même API qui réserve d'accès aux images et aux métadonnées du fac-similé numérique et qui est discuté dans la section 7.2.1. L'application `alto2tei` permet de personnaliser la requête envoyée à l'API IIIF selon les normes de l'institution qui le gère. Le fichier configuré pour l'API IIIF géré par la BnF est montré dans la Figure 7.7. Les paramètres `manifest_prefix` et `image_prefix` sont

pareils pour l'API IIIF de la BnF, mais le fichier de configuration les garder distinct afin qu'ils puisse s'adapter aux APIs qui distinguent entre l'URL de l'IIIF Image API et du Presentation l'API.

```
iiifURI:
  scheme: "https"
  server: "gallica.bnf.fr"
  manifest_prefix: "/iiif/ark:/12148/"
  image_prefix: "/iiif/ark:/12148/"
  manifest_suffix: "/manifest.json"
```

FIGURE 7.7 – La partie du fichier de configuration portant sur la requête envoyée à l'API IIIF

Enfin, le fichier de configuration informe les éléments descendant du `<publicationStmt>` dans le `<fileDesc>` du `<teiHeader>`. Les métadonnées récupérées par l'application `alto2tei` ainsi que leur format en YAML sont montrés dans la Figure 7.8. Le nom du projet (ligne 18, Fig. 7.8) et les noms des membres de l'équipe peuvent être personnalisés. Le prénom ou les prénoms de chaque membre sont renseignés par la chaîne entre guillemets qui suit les deux points après le terme *forename*. Le terme *surname* indique le même chose pour le nom de famille. L'identifiant du membre peut être personnalisé soit en le type d'identifiant, tel que ORCID, soit en la valeur qui suit les deux points après le terme *target*.

```
responsibility:
  text: "Transformation from ALTO4 to TEI by"
  resp: [{"forename":"Kelly",
    "surname":"Christensen",
    "ptr":{"type":"orcid","target":"000000027236874X"}},

    {"forename":"Simon",
    "surname":"Gabay",
    "ptr":{"type":"orcid","target":"0000000190944475"}},

    {"forename":"Ariane",
    "surname":"Pinche",
    "ptr":{"type":"orcid","target":"0000000278435050"}}
    ]
  # name the publisher of this digital edition (project name)
  publisher: "Gallic(orpor)a"
  # name which authority finances or supports the project
  authority: "BnF DATALab"
  # specify the edition's availability to the public/licencing
  availability: {"status": "restricted", "n":"cc-by"}
  licence: {"target":"https://creativecommons.org/licenses/by/4.0/"}
```

FIGURE 7.8 – La partie du fichier de configuration portant sur la requête envoyée à l'API IIIF

## 7.3 La mise en place des données récupérées

Ayant découvert les structures de données des quatre sources de données externes, il faut parler d'où ses données ainsi récupérées vont dans le `<teiHeader>`. Les métadonnées intégrées au document TEI s'appuient sur (1) le *manifest* IIIF envoyé par l'API IIIF en format JSON, (2) les données UNIMARC du catalogue général de la BnF envoyées par l'API SRU en format XML, (3) les données HTML récupérées à partir de la page de recherche du site SUDOC, et (4) le fichier de configuration en format YAML. L'application recherche des données dans deux autres sources : (5) les fichiers ALTO et (6) la commande qui démarre l'application. L'application `alto2tei` compte les fichiers ALTO du fac-similé numérique et utilise ce nombre pour l'élément `<extent>`. De plus, elle s'appuie sur la commande saisie pour démarrer l'application. La commande exige un paramètre qui déclare la version du engin *Kraken* qui a été utilisé pour mettre en œuvre les modèles HTR.

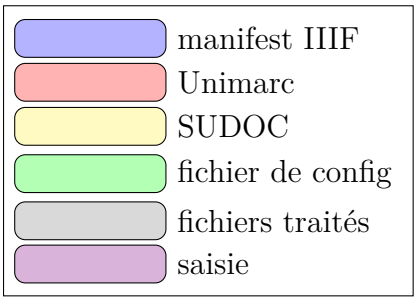
### 7.3.1 Les sources des données du `<titleStmt>`

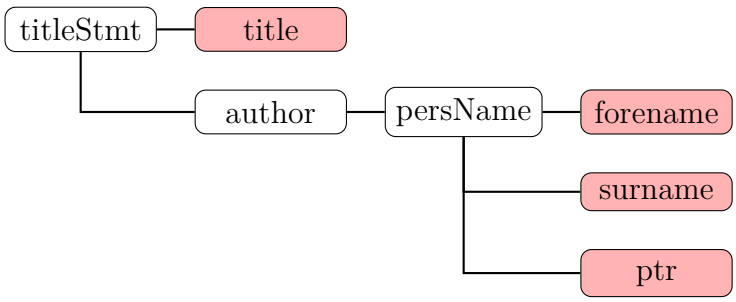
La Figure 7.9 visualise les sources des données du `<titleStmt>`. Les éléments portant sur le titre de la ressource (`<title>`) et sur les personnes auxquels est attribuée la propriété intellectuelle du texte représenté (`<author>`) sont informés par l'un de deux sources de données possible. Soit les données du catalogue général donnent beaucoup d'information sur les auteurs ainsi qu'un titre propre, soit le *manifest* IIIF donne le titre du texte et les noms non annotés des auteurs. Le choix entre les deux sources de données se fait à partir de la disponibilité des données du catalogue de la BnF. D'habitude, l'application `alto2tei` met en priorité les données du catalogue. Les éléments descendants du `<respStmt>` s'appuient toujours sur le fichier de configuration afin qu'ils puissent être personnalisés par les personnes qui utilisent l'application `alto2tei` pour générer leurs propres documents TEI.

La Figure 7.9a visualise les éléments du `<titleStmt>` dans le cas où la relation entre le fac-similé numérique sur Gallica et le document physique appartenant à la BnF est établie. Cela veut dire que le document dont parle la notice du catalogue est le même document à partir duquel le fac-similé numérique sur Gallica a été fait. Quand la notice du bon document est trouvée, les données UNIMARC cherchées depuis le catalogue sont (1) le titre uniforme du document et (2) le nom et l'identifiant de ses auteurs.

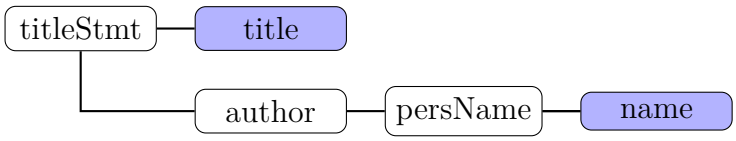
### 7.3.2 Les sources des données du `<extent>`

L'élément `<extent>` du `<teiHeader>`, qui indique le nombre de fichiers traités par l'application `alto2tei` lors de la création de la ressource numérique, s'appuie simplement sur un compte des fichiers. L'application a besoin de traiter tout fichier ALTO

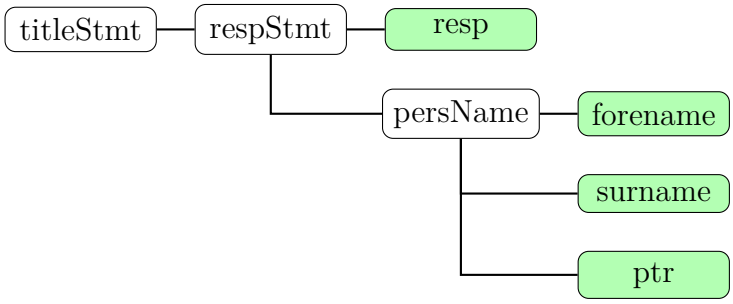



(a) Les sources de données du `<title>` et du `<author>` quand la relation entre le fac-similé numérique et le document physique est établie




(b) Les sources de données du `<title>` et du `<author>` quand la relation entre le fac-similé numérique et le document physique *n'est pas* établie




(c) Les sources de données du `<respStmt>` dans tout cas

FIGURE 7.9 – Les sources de données des éléments du `<titleStmt>`

dans le dossier du document, expliqué dans la section 4.1.1, et mettre ses données dans le `<sourceDoc>`, selon la modélisation de l'application `alto2tei`. Elle peut également compter le nombre des fichiers ALTO qui appartiennent au document. Ce compte elle met comme la « taille » de la ressource numérique, dans l'élément `<measure>`.

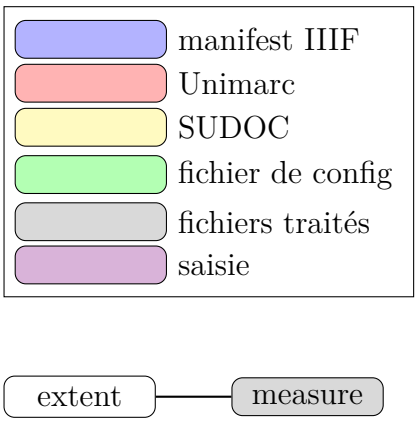



FIGURE 7.10 – La source des données du `<extent>`

### 7.3.3 Les sources des données du `<publicationStmt>`

La distribution de la ressource numérique est décrite par l'élément `<publicationStmt>`. Cet élément peut compter uniquement–et idéalement–sur les données du catalogue général de la BnF en format UNIMARC. Mais si l'application a besoin, elle peut remplir certains éléments du `<publicationStmt>` en laissant vide des autres puisque le *manifest* IIIF peut renseigner sur l'auteur du document, son titre, et sa date de création. Le *manifest* IIIF n'est pas une source sûre sur l'éditeur ni sur le lieu de publication. Les données du catalogue général de la BnF, par contre, portent toujours sur ces détails. Même si elles disent que la donnée n'est pas connue, la notice du catalogue général renseigne toujours sur l'éditeur et le lieu de publication du document.

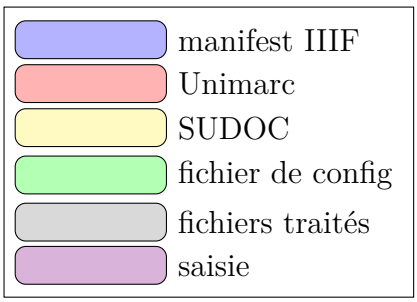



FIGURE 7.11

### 7.3.4 Les sources des données du `<sourceDesc>`

Le `<sourceDesc>` décrit le document source dont la transcription et le texte la ressource numérique représente. Il se compose de deux éléments principales. Dans un premier temps, il contient le `<bibl>` qui est une description de la source, en parlant de ses auteurs, de son titre, de l'éditeur, et de sa date et lieu de création. Dans un deuxième temps, le `<sourceDesc>` contient le `<msDesc>` qui est une citation bibliographique du document source. Les descendants du `<sourceDesc>` comptent sur le plus grand nombre de sources de données par rapport aux autres éléments du `<fileDesc>` puisque la description du document source (`<bibl>`) et la citation bibliographique du document (`<msDesc>`) s'appuient tous les deux sur le document physique et sur le fac-similé numérique.

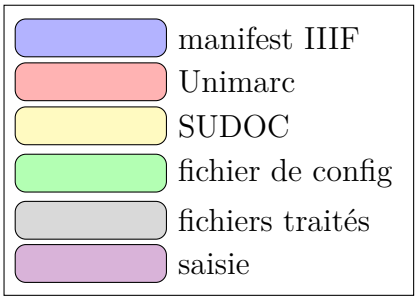



FIGURE 7.12

### 7.3.5 Les sources des données du `<profileDesc>`

Le `<profileDesc>` de la ressource numérique généré par l'application `alto2tei` est minimal est ne présente que de la donnée sur la langue utilisé dans le document source. En outre, la version actuelle de l'application ne supporte qu'une seule langue. Malgré la simplicité conceptuelle du `<profileDesc>`, son élément `<language>` compte sur deux sources de données. Le mot balisé dans l'élément vient du *manifest* IIIF puisque la langue est l'une des données déterminée essentielle et recherchée lors du traitement du *manifest*. La valeur de l'attribut `@ident` dans l'élément `<language>` est récupérée depuis le catalogue général de la BnF parce qu'elle est une clef standardisé pour la langue utilisée.

### 7.3.6 Les sources des données du `<encodingDesc>`

La description de l'encodage et de l'appareil qui a fait la transcription sont essentiels. Sans eux, la transcription présentée dans la ressource ne pourra pas être reproduite. Et sans pouvoir reproduire la transcription, les scientifiques ne peuvent pas la vérifier. Pouvoir reproduire des résultats scientifiques est fondamental à la recherche.

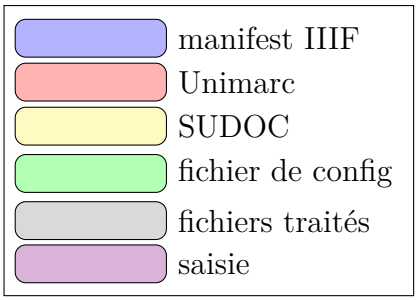
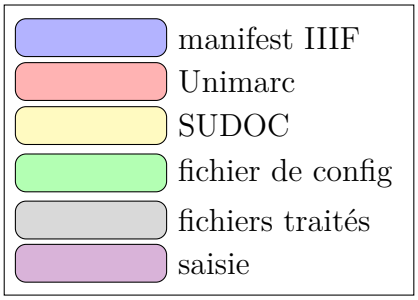


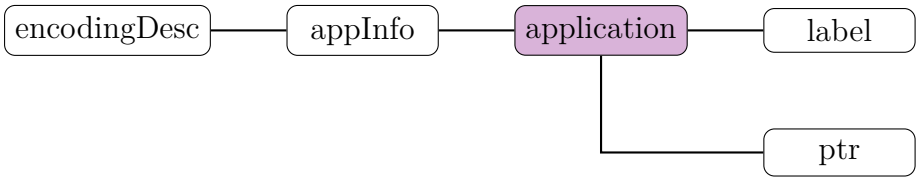

FIGURE 7.13

Le `<encodingDesc>` contient deux descendants directs. Dans un premier temps, il contient des informations sur l'engin qui a mis en œuvre les modèles HTR (`<application>`). L'application `alto2tei` actuelle supporte seulement l'engin *Kraken*. L'élément `<appInfo>` est créé toujours pour les versions de *Kraken*. Mais quand l'application `alto2tei` est démarré depuis la ligne de commande, un paramètre obligatoire déclare quelle version de *Kraken* a été utilisé pour créer les fichiers ALTO. L'application `alto2tei` tient ce paramètre et l'utilise pour renseigner sur la version de l'application *Kraken* dans l'élément `<application>`.

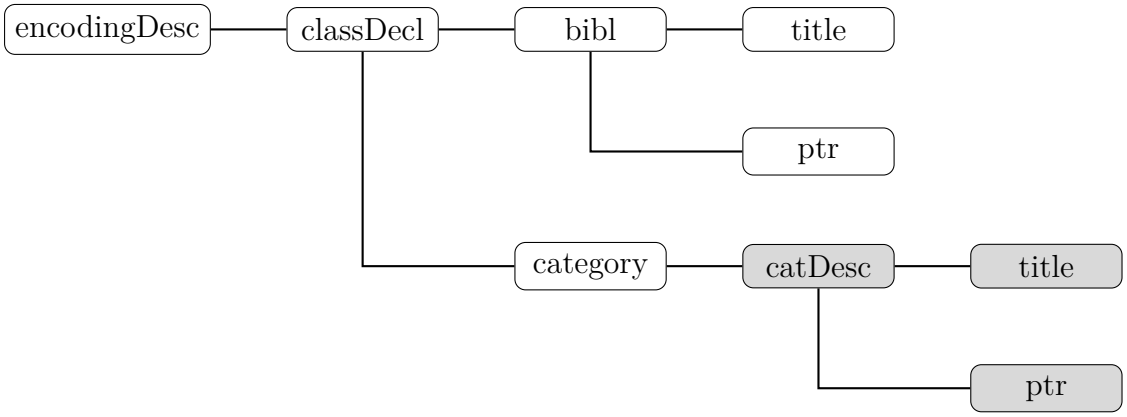
Dans un deuxième temps, le `<encodingDesc>` s'appuie sur les fichiers ALTO que l'application `alto2tei` traite. L'application analyse les étiquettes de ligne et de zone que les modèles HTR ont appliquées. Ensuite, elle compare les étiquette trouvées avec une liste des étiquettes du vocabulaire *SegmOnto*. Pour chaque type d'étiquette utilisé dans les fichiers ALTO qui appartient au vocabulaire *SegmOnto*, l'application *alto2tei* la déclare dans la bonne catégorie de l'élément `<taxonomy>` du `<classDecl>`. Un exemple d'une telle liste des étiquettes utilisés dans des fichiers ALTO est visualisé dans la Figure 7.14b.

En outre, puisque le projet *Gallic(orpor)a* a choisi de s'appuyer sur le vocabulaire *SegmOnto* (cf. Chap. 2), certains éléments du `<classDecl>` sont encodés en dur. La taxonomie déclarée dans l'identifiant de l'élément `<taxonomy>` ainsi que le titre déclaré (`<title>`) est toujours « SegmOnto ». De plus, le *pointer* (`<ptr>`) qui indique l'URL où se trouvent des informations mises à jour sur le vocabulaire utilisé est toujours celui du projet *SegmOnto*. Ces données ne sont pas personnalisables. En plus, bien que les descendants des catégories (`<category>`) des lignes et des zones soient informés par les fichiers ALTO, les données encodées dans les éléments viennent plutôt d'une liste que l'application `alto2tei` maintient. Cette liste encodée en dur porte l'URL de toute zone et toute ligne sur le site web du projet *SegmOnto*. L'application `alto2tei` récupère l'étiquette *SegmOnto* utilisée dans des fichiers ALTO. Elle nettoie cette donnée et la met dans l'élément `<title>`, puis elle la enrichit avec l'URL qu'elle tient dans sa liste et met cette dernière donnée dans l'élément `<ptr>`.




(a) La source de données du `<appInfo>`




(b) La source de données du `<classDecl>`

FIGURE 7.14 – Les sources de données du `<encodingDesc>`

## 7.4 Le *mapping* des toutes données du <teiHeader>

Le *mapping* complet des données du <teiHeader> est visualisé dans la Table ?? ci-dessous. La table montre en détail la localisation de chaque donnée cherchée, dont les sources sont décrites dans les sections précédents de ce chapitre. Les données UNIMARC du catalogue général de la BnF et les données HTML du site SUDOC, étant en format XML, exigent plus de détail pour se localiser. Par exemple, l'élément ou l'attribut qui contient la donnée cherchée ne suffit pas puisqu'il en y a souvent plusieurs du même nom dans l'arbre XML. La Table ?? décrit donc la localisation des données en listant les éléments ou les composants qui précèdent la donnée cherchée. Pour les attributs d'un élément XML, la table utilise le syntaxe *XPath* par lequel l'attribut est déclaré avec l'arobase. Par exemple, l'attribut @att d'un élément <element>, qui descende de l'élément <parent>, serait représenté dans le format suivant : parent/element[@att]. Les données en JSON et YAML du *manifest* IIIF et du fichier de configuration, respectivement, n'exigent pas une telle solution parce que les données du *manifest* viennent toutes de la clef *metadata* et le fichier de configuration ne répète pas des étiquettes.

# Chapitre 8

# La génération du <sourceDoc>

À partir d'une image numérique, des modèles HTR produisent une transcription de la page en format XML ALTO (Analyzed Layout and Text Object). Comme expliqué et justifié dans le chapitre 4, ce format est transformé en le format *pivot* du pipeline *Gallic(orpor)a*, la TEI (Text Encoding Initiative). Le schème TEI est flexible et il permet d'encoder la transcription d'un document texte en plusieurs façons. La manière préférée par l'équipe du projet *Gallic(orpor)a* s'appuie sur l'élément TEI `<sourceDoc>`. Selon les *guidelines* de la Text Encoding Initiative, l'élément `<sourceDoc>` contient *a transcription or other representation of a single source document potentially forming part of a dossier génétique or collection of sources*[1]. L'autre option est l'élément `<facsimile>`. Il contient *a representation of some written source in the form of a set of images rather than as transcribed or encoded text*[2].

Entre les deux options, l'élément `<souceDoc>` convient mieux à la transcription encodée dans un fichier ALTO puisqu'il est destiné à traiter la représentation d'une source. Un fichier ALTO contient la représentation d'une image de texte. L'élément `<facsimile>` pourrait bien servir à l'encodage de l'image elle-même, mais sa représentation dans le fichier ALTO est mieux encodée avec l'élément `<sourceDoc>`. Après tout, il faut se souvenir que le pipeline *Gallic(orpor)a* vise à conserver dans le document TEI toute donnée de la transcription du fichier ALTO. L'un des objectifs du projet est que la ressource numérique produite par le pipeline permet de recréer des fichiers ALTO à partir du document TEI, afin que des utilisateurs puissent utiliser les fichiers ALTO reconstruits comme des vérités de terrain et entraîner des nouveaux modèles HTR. Il faut donc un élément TEI destiné à la transcription d'une image, au lieu de l'image elle-même.

Par rapport au *mapping* des données du `<teiHeader>`, le *mapping* des données du `<sourceDoc>` est plus direct et fixé. Tandis que le contenu du `<teiHeader>` compte sur la disponibilité variable des données depuis les divers sources de données, le contenu du

1. *TEI element sourceDoc.*
2. *TEI element facsimile.* URL : `https://tei-c.org/release/doc/tei-p5-doc/en/html/ref-facsimile.html` (visité le 03/09/2022).

<sourceDoc> est très prévisible et compte sur un schème ALTO qui est très systématique. La manière de transcrire les pages numérisées ne change pas et s'effectue par les mêmes modèles HTR bien que les documents transcrits soient différents de l'un à l'autre.

Cependant, il y a une variation possible dans la granularité de la transcription. D'une part, le fichier ALTO peut porter des prédictions sur les caractères d'un mot, y compris leur contenu et leur emplacement sur la page. De l'autre part, le fichier ALTO peut porter des prédictions sur une ligne de texte, où son contenu est une chaîne des mots et des espaces entre mots. Dans ce dernier cas plus simple, le fichier ALTO présente moins de détail par rapport à l'autre forme. L'application `alto2tei` s'adapte aux deux puisqu'elles sont toutes les deux valables et produites par l'engin *Kraken*. Depuis la ligne de commande, l'engin *Kraken* met en pratique des modèles HTR et produit les fichiers ALTO qui contiennent des prédictions sur les caractères ou les « glyphes » des mots de la ligne de texte. Depuis l'interface *eScriptorium*, la même version de *Kraken* et les mêmes modèles HTR produisent des fichiers ALTO qui ne contiennent que des prédictions sur la ligne de texte, n'allant pas en détail sur les caractères et les mots ou les « segments ». Les formes différentes sont expliquées dans la section 5.1.2 du chapitre 5.

## 8.1 Le modèle du <sourceDoc>

L'application `alto2tei` récupère toute donnée significative des fichiers ALTO et les met sur un arbre TEI qu'elle construit, spécifiquement sur la branche de l'élément <sourceDoc>. L'arborescence du <sourceDoc> prendre deux formes, selon le détail du fichier ALTO que des modèles HTR ont produit. Si les modèles avaient enregistré des prédictions sur les glyphes et les segments dans le fichier ALTO, le <sourceDoc> aurait quatre niveaux d'élément <zone> pour porter sur les masques d'un bloc de texte, d'une ligne dans le bloc, d'un mot dans la ligne, et d'un caractère dans le mot. Le contenu textuel serait donc représenté deux fois ; il serait balisé dans l'élément qui porte sur la prédiction du caractère et aussi dans un élément qui représente tous caractères prédits dans une ligne de texte. L'exemple des quatre niveaux est montré dans la Figure 8.1 Si les modèles avaient enregistré uniquement des prédictions sur les blocs et les lignes de texte, le <sourceDoc> aurait deux niveaux d'élément <zone>, un pour représenter le bloc prédit et l'autre pour la ligne prédite. L'exemple de cette arborescence plus simple est montré dans la Figure 8.2.

### 8.1.1 La page

Normalement, un document XML ALTO représente la transcription d'une seule page du document source. Dans le schème ALTO, des données portant sur une page sont imbriquées dans l'élément <Page> dont les attributs décrivent le longueur (@WIDTH) et

```
<sourceDoc>
  <surface><!-- Région d'une page -->
<!-- ... -->
    <zone type="SegmOntoZone"><!-- Région d'un bloc de texte -->
<!-- ... -->
      <zone type="SegmOntoLine"><!-- Région d'une ligne de texte (ex. "
    Texte ici.") -->
<!-- ... -->
        <zone type="String"><!-- Région d'un segment dans la ligne (ex. "
    Texte") -->
<!-- ... -->
          <zone type="Glyph"><!-- Région d'un caractère dans le segment (ex
    . "T") -->
<!-- ... -->
            <c>T</c>
          </zone>
<!-- ... -->
        <zone type="Space"/><!-- Région d'une espace entre mots dans la
    ligne -->
<!-- ... -->
        <line>Texte ici.</line>
      </zone>
    </zone>
  </surface>
</sourceDoc>
```

FIGURE 8.1 – Le `<sourceDoc>` de quatre niveaux de masques imbriqués

l'hauteur (`@HEIGHT`), ainsi que le compte d'image (`@PHYSICAL_IMAGE_NR`) dans la suite des images traitées. Contrairement aux données susdites, il ne faut pas conserver l'identifiant (`@ID`) donné à la page. En reconstituant un fichier ALTO à partir du `<sourceDoc>`, peu important quel identifiant peut être donné à la nouvelle `<Page>` pourvu qu'il soit unique. L'application `alto2tei` génère un nouveau identifiant pour la `<Page>` et pour tout élément TEI qu'elle construit.

### Des règles générales sur l'identifiant de l'élément TEI

L'identifiant de tout élément descendant du `<sourceDoc>` se construit de certains composants qui s'accumulent. Le parent, la page, porte tout simplement l'identifiant de la page, c'est-à-dire le numéro du folio. Tout élément descendant et imbriqué dans la page retient cette donnée dans son identifiant, en y ajoutant à la suite encore plus de données. Par exemple, l'identifiant de la page de l'onzième folio du fac-similé numérique serait « f11 ». L'identifiant du premier bloc du texte sur l'onzième folio porterait donc l'identifiant « f11-textblock_0-blockCount1 ». L'identifiant du bloc se compose de l'identifiant de la page, l'identifiant donné au premier élément `<TextBlock>` et enfin une traduction de ce dernier composant en « blockCount1 » qui commence compter les blocs à partir du numéro 1 au lieu d'à partir de zéro, comme font souvent les logiciels HTR. Pour donner encore un exemple, l'identifiant de la première ligne de texte du premier block sur l'onzième folio

```
<sourceDoc>
  <surface><!-- Région d'une page -->
<!-- ... -->
    <zone type="SegmOntoZone"><!-- Région d'un bloc de texte -->
<!-- ... -->
      <zone type="SegmOntoLine"><!-- Région d'une ligne de texte (ex. "
    Texte ici.") -->
        <line>Texte ici</line>
      </zone>
    </zone>
  </surface>
</sourceDoc>
```

FIGURE 8.2 – Le `<sourceDoc>` de deux niveaux de masques imbriqués

porterait l'identifiant « f11-textblock_0-textline_0-lineCount1 ». Encore, l'identifiant se compose des étiquettes des éléments parents (f11, textblock_0) ainsi qu'une traduction du dernier composant (textline_0) en une chaîne plus logique (lineCount1).

#### La page en particulier

Selon notre modélisation, l'élément TEI `<surface>` représente une page et contient donc toute donnée encodée dans l'élément ALTO `<Page>` et son enfant `<PrintSpace>`. Souvent le fichier ALTO présente le longueur et l'hauteur de la page dans les éléments `<Page>` et `<PrintSpace>`. Cet redondance se produit au niveau de la page parce que l'entièreté de la page traitée est aussi ce qui est transcrit. Quand on construit un fichier ALTO à partir du document TEI il faut recréer cette redondance en répétant le longueur et l'hauteur dans les deux éléments ALTO. La transformation en TEI représente ces deux données dans un seul élément, le `<surface>`. Imbriqué dans l'élément `<surface>`, l'élément `<graphic>` combine l'ARK du fac-similé numérique et l'URI IIIF pour l'IIIF Image API de la BnF. Cet URI renvoie l'image entière de la page numérisée. L'exemple du `<Page>` et l'exemple de sa modélisation en TEI sont donnés dans la Figure 8.3. Une visualisation de la transformation d'ALTO à TEI est donnée à la fin de ce chapitre, dans la Figure 8.8.

### 8.1.2 Le bloc

La *SegmOntoZone* indique un bloc sur la page. Elle peut contenir du texte, comme dans le cas d'une *MainZone*, ou elle peut n'en avoir pas, comme dans le cas d'une *GraphicZone* qui décrit la région d'une page dans laquelle se trouve un dessin. Un fichier ALTO encode tout type de bloc dans l'élément `<TextBlock>` même s'il ne contient pas du texte. Étant modélisées en TEI, les données du `<TextBlock>` sont transformées en l'élément `<zone>`.

```
<Layout>
  <Page WIDTH="2568" HEIGHT="3631" PHYSICAL_IMG_NR="2" ID="page_2">
    <PrintSpace HPOS="0" VPOS="0" WIDTH="2568" HEIGHT="3631">
<!-- ... -->
    </PrintSpace>
  </Page>
</Layout>
```

(a) Le <Page> en ALTO

```
<surface xml:id="f11" n="2" ulx="0" uly="0" lrx="2568" lry="3631">
  <graphic url="https://gallica.bnf.fr/iiif/ark:/12148/btv1b8610802d/f11/
    full/full/0/native.jpg"/>
<!-- ... -->
</surface>
```

(b) Le <surface> en TEI

FIGURE 8.3 – La modélisation du <Page>

### Des règles générales pour l'élément <zone> dans le modèle TEI

Avant d'aller en plus de détail particulier à la transformation du <TextBlock> en TEI, il faut parler de certaines transformations généralisées pour plusieurs éléments du fichier ALTO. Dans notre modélisation, l'élément TEI <zone> représente plus que le <TextBlock> du schème ALTO. En fait, il représente tout masque prédit par des modèles HTR. La région prédite d'un bloc (*SegmOntoZone*) et celle d'une ligne de texte (*SegmOntoLine*) ainsi que celle d'un mot et celle d'un glyphe sont toutes représentées par l'élément TEI <zone>. Il est bien adapté à représenter les données géométriques d'un masque.

Le <zone> doit porter certains attributs d'usage, peu importe quel type de masque il représente. Ces attributs décrivent l'étiquette (@type) appliquée à la région décrite et les quatre coordonnées (@HPOS, @VPOS, @WIDTH, @HEIGHT) du rectangle qui l'encadre. Comme explique la section 5.1.2, les valeurs des attributs @HPOS et @VPOS font les coordonnées x et y, respectivement, du point le plus haut à gauche du rectangle, comme se voit dans la Figure 5.2. La valeur de l'attribut @HEIGHT compte la différence entre le point le plus haut et le point le plus bas du rectangle. La valeur de l'attribut @WIDTH calcule aussi la différence entre le côté gauche du carré et son côté droit. En outre, les quatre coordonnées du rectangle se sont transformés afin de construire l'attribut @source pour tout <zone>. Le @source fournit l'URL pour visionner la région de l'image dans un API IIIF. Selon les normes de l'IIIF, l'URL se compose des parties suivantes :

| titre | exemple |
|---|---|
| *scheme* | https :// |
| *server* | gallica.bnf.fr |
| *prefix* | /iiif/ark :/12148 |
| *identifier* (/ARK/folio) | /btv1b8610802d/f11 |
| nombre de pixels entre la position 0 et la position la plus à gauche de la région sur l'axe des x (`@HPOS` en ALTO) | 323 |
| nombre de pixels entre la position 0 et la position la plus en haute de la région sur l'axe des y (`@VPOS` en ALTO) | 336 |
| nombre de pixels entre la position la plus à gauche et celle la plus à droite sur l'axe des x (`@WIDTH` en ALTO) | 2056 |
| nombre de pixels entre la position la plus en haute et celle la plus en bas sur l'axe des y (`@HEIGHT` en ALTO) | 2812 |
| *size* | full |
| *rotation* | 0 |
| *quality* | native |
| *.format* | .jpg |

Les composants de la table ci-dessus constituent l'URL suivant :

```
https://gallica.bnf.fr/iiif/ark:/12148/btv1b8610802d/f11
    /323,336,2056,2812/full/0/native.jpg
```

Cet URL se donne comme la valeur de l'attribut `@source` des éléments `<zone>` dans notre modélisation TEI. Il permet de visionner la partie du fac-similé numérique concernée depuis un éditeur, tel que *TEIPublisher*, qui requête l'image de l'API IIIF.

Normalement, les modèles HTR d'aujourd'hui prédisent le rectangle qui encadre la région sur la page et aussi le polygone qui fait un masque plus précis. Si le modèle produit les deux formes de masque, il les font uniquement pour les régions sur la page qui contiennent soit du texte, soit une image. Dit autrement, tout type de région, y compris l'espace entre mots, s'encadre dans un rectangle, mais les types qui contiennent quelque chose autre qu'une espace vide, donc toute région sauf l'espace entre mots, s'encadrent dans un polygone. Le polygone porte plus de coordonnées que le rectangle. Dans le schème ALTO, les valeurs des coordonnées du polygone sont données dans l'attribut `@POINTS` de l'élément `<Polygon>` qui descend indirectement de l'élément sur lequel il porte. Notre modélisation en TEI représent les coordonnées du polygone dans l'attribut `@points` du même élément `<zone>` qui est concerné.

### Le bloc (<TextBlock>) en particulier

En plus des données d'usage (le type et les coordonnées du polygone et/ou du rectangle) la modélisation TEI du <TextBlock> exige la composition d'URL pour visionner le masque du bloc (@source) et la décomposition de l'étiquette appliquée au bloc. Entraîné sur le vocabulaire *SegmOnto*, le modèle HTR devrait donner au bloc une référence à une étiquette qui peut se composer de trois parties : le type, le sous-type, et le numéro dans la suite. Le deuxième colonne prédite sur la page, par exemple, porterait l'étiquette *MainZone :column :2*. Les étiquettes ainsi composées sont données aux blocs et aux lignes de texte. Les parties du document encore plus petit, tel que le mot ou le glyphe, ne portent pas d'étiquette composée.

Par conséquent, uniquement les étiquettes attribuées aux éléments ALTO <TextBlock> et <TextLine> sont décomposées lors de leur transformation TEI. Elles peuvent se diviser en trois. L'attribut @type prend la première partie, le @subtype prend le sous-type qui pourrait suivre les deux points, et le @n prend le numéro s'il y en a un qui suit les deux points à la fin. Si le modèle HTR n'a pas mis en pratique des étiquettes aussi détaillées, les attributs @subtype et @n prennent la valeur *none* pour le bloc. Mais pour la ligne de texte (<TextLine>), la valeur du numéro se constitue du compte que fait l'application alto2tei des lignes de texte traitées sur la page. Un tel compte n'est pas si logique pour les blocs et donc l'attribut @n ne porte pas de valeur significative si l'étiquette n'en a pas donnée aucune. L'exemple du <TextBlock> et l'exemple de sa modélisation en TEI sont donnés dans la Figure 8.4. Une visualisation de la transformation d'ALTO à TEI est donnée à la fin du chapitre dans la Figure 8.9.

```
<TextBlock HPOS="323" VPOS="336" WIDTH="2056" HEIGHT="2812" ID="textblock_0
    " TAGREFS="BT2062">
  <Shape>
    <Polygon POINTS="2379 336 2379 3148 323 3148 323 336"/>
  </SHAPE>
<!-- ... -->
</TextBlock>
```

(a) Le <TextBlock> en ALTO

```
<zone xml:id="f11-textblock_0-blockCount1" type="MainZone" corresp="#
    MainZone" subtype="none" n="none" ulx="323" uly="336" lrx="2379" lry="
    3148" points="379,336 2379,3148 323,3148 323,336" source="https://
    gallica.bnf.fr/iiif/ark:/12148/btv1b8610802d/f11/323,336,2056,2812/full
    /0/native.jpg">
<!-- ... -->
</zone>
```

(b) Le <zone> du <TextBlock> en TEI

FIGURE 8.4 – La modélisation du <TextBlock>

### 8.1.3 La ligne de texte

Dans le vocabulaire *SegmOnto*, l'étiquette *line* s'applique à la région de l'image dans laquelle s'encadre une ligne de texte. L'élément ALTO qui prend cette donnée est l'élément `<TextLine>`. Contrairement au `<TextBlock>` qui ne contient pas forcement du texte, l élément `<TextLine>` doit avoir des prédictions du texte encodées dedans et doit donc avoir d'enfants qui descendent de lui. Comme l'étiquette *SegmOnto* du `<TextBlock>`, celle du `<TextLine>` se divise en trois parties. Si la valeur du `@type`, la première partie de l'étiquette, est identique à l'une des étiquettes listée dans le `<taxonomy>` du `<teiHeader>`, l'élément portera aussi l'attribut `@corresp` qui prendra comme valeur une référence à la classe.

La modélisation des données du `<TextLine>` en TEI s'appuient comme d'habitude sur l'élément `<zone>` parce qu'il porte sur la représentation d'une région de la page et une partie des données du `<TextLine>` portent sur le masque de la ligne. Les attributs du `<zone>` pour la ligne de texte sont identiques à ceux du bloc de texte. Il y a les quatre coordonnées du rectangle `@ulx`, `@uly`, `@lrx`, `@lry` récupérées respectivement depuis les attributs suivants du `<TextLine>` : `@HPOS`, `@VPOS`, `@WIDTH`, `@HEIGHT`. Ensuite l'attribut `@source` se compose en part de ces quatre coordonnées. Enfin, le `<zone>` contient les points du `<Polygon>`, l'élément qui descend du `<TextLine>` et qui décrit le périmètre du polygone qui encadre la ligne de texte.

Le `<zone>` du `<TextLine>` contient deux enfants directs particulier à la ligne de texte : le `<line>` et le `<path>`. L'élément `<line>` contient le texte de la ligne. Comme attribut, il porte simplement un identifiant et le nombre de la ligne de texte lors du traitement du fichiers ALTO. Le `<path>` représente le *baseline* de la ligne de texte, c'est-à-dire le début et la fin de la ligne linéaire. Il se compose donc de quatre nombres, un pair x,y indiquant le point du début et un deuxième pair x,y indiquant le point de la fin. Les quatre nombres sont encodés directement dans le fichier ALTO comme la valeur de l'attribut `@BASELINE` du `<TextLine>`. L'élément TEI qui convient le mieux à la donnée du *baseline* est l'élément `<path>`. L'exemple du `<TextLine>` et l'exemple de sa modélisation en TEI sont donnés dans la Figure 8.5. Une visualisation de la transformation d'ALTO à TEI est donnée dans la Figure 8.10.

#### La ligne de texte quand il y a des prédictions sur les mots et les glyphes dedans

Pour certains fichiers ALTO, tel que ceux qui sortent de l'interface *eScriptorium*, la modélisation en TEI s'arrête là. Le fichier ALTO ne porte pas de plus de détail après la ligne de texte. Mais pour certains d'autres fichiers, tel que ceux qui sont produits par l'engin *Kraken* depuis la ligne de commande, ils attestent aux prédictions sur des mots et sur des glyphes. Dans ce cas, la ligne de texte a plus de descendants, mais afin de garder une arborescence générique qui supporte des comparaisons entre des fichiers de divers

```
<TextLine ID="textline_0" TAGREFS="LT722" BASELINE="605 944 2010 925" HPOS=
    "596" VPOS="777" WIDTH="1414" HEIGHT="182">
  <Shape>
    <Polygon POINTS="605 944 596 816 666 795 669 795 672 795 814 810 838
    792 838 789 841 789 844 789 847 789 932 804 953 789 956 789 959 789 962
    789 1050 801 1323 783 1326 783 1704 798 1768 777 1771 777 1774 777 2004
    798 2010 925 2004 953 1798 941 1750 956 1747 956 1744 956 605 959"/>
  </Shape>
  <String CONTENT="A MONSIEVR" HPOS="596" VPOS="777" WIDTH="1414" HEIGHT="
    182"/>
</TextLine>
```

(a) Le <TextLine> en ALTO où le texte est contenu dans l'attribut @CONTENT de l'élément descendant <String>

```
<zone xml:id="f11-textblock_0-textline_0-lineCount1" type="HeadingLine"
    corresp="#HeadingLine" subtype="none" n="none" ulx="596" uly="777" lrx="
    2010" lry="959" points="605,944 596,816 666,795 669,795 672,795 814,810
    838,792 838,789 841,789 844,789 847,789 932,804 953,789 956,789 959,789
    962,789 1050,801 1323,783 1326,783 1704,798 1768,777 1771,777 1774,777
    2004,798 2010,925 2004,953 1798,941 1750,956 1747,956 1744,956 605,959"
    source="https://gallica.bnf.fr/iiif/ark:/12148/btv1b8610802d/f11
    /596,777,1414,182/full/0/native.jpg">
  <path xml:id="f11-textblock_0-textline_0-lineCount1-baseline" points="
    605,944 2010,925"/>
  <line xml:id="f11-textblock_0-textline_0-lineCount1-text" n="1">A
    MONSIEVR</line>
</zone>
```

(b) Le <zone> du <TextLine> en TEI

FIGURE 8.5 – La modélisation du <TextLine>

formats, le <zone> de la ligne de texte garde toujours les mêmes deux enfants : le <path> et le <line>. En plus de ces deux, le <zone> contient une suite d'éléments <zone> pour tout segment prédit sur la ligne, soit un mot, soit une espace entre mots.

### 8.1.4 Le segment

Les fichiers ALTO qui contiennent plus de détail ont des éléments <zone> pour des segments (<String>) prédits et pour les glyphes (<Glyph>) prédits dans les segment. Les modèles HTR prédisent les segments qui contiennent des glyphes, tel qu'un mot, et ceux qui n'en ont pas, tel qu'une espace entre mots. Les segment qui contiennent des glyphes peuvent représenter la prédiction d'un mot, de la ponctuation, ou d'un mot avec de la ponctuation à côté. L'important est que le segment contient soit la prédiction d'au moins un glyphe ou la prédiction d'une espace entre mots. Uniquement les segments (<String>) qui contiennent des prédictions sur des glyphes portent aussi un polygone dans lequel s'encadre la chaîne des glyphes. Si le <String> représente une espace entre

mots, le masque prédit n'est qu'un rectangle. Bien que l'élément <String> ne porte pas directement sur le texte prédit, le modèle HTR évalue son taux de réussite à partir des glyphes bien prédits dedans. L'évaluation de sa prédiction est représentée dans l'attribut @WC de l'élément <String>. L'acronyme *WC* signifie en anglais *word confidence*. L'exemple du <String> et l'exemple de sa modélisation en TEI sont donnés dans la Figure 8.6. Une visualisation de la transformation d'ALTO à TEI est donnée dans la Figure 8.11.

```
<String ID="segment_1" CONTENT="MONSIEVR" HPOS="837" VPOS="777" WIDTH="1172
    " HEIGHT="182" WC="0.9.64">
  <Shape>
    <Polygon POINT="..."/>
  </Shape>
<!-- ... -->
</String>
```

(a) Le <String> en ALTO où le texte n'est pas contenu dans l'attribut @CONTENT de l'élément descendant <String>

```
<zone xml:id="f11-textblock_0-textline_0-segment_2-segCount3" type="String"
    ulx="837" uly="777" lrx="2009" lry="959" points="..." source="https://
    gallica.bnf.fr/iiif/ark:/12148/btv1b8610802d/f11/837,777,1172,182/full
    /0/native.jpg">
  <certainty xml:id="f11-textblock_0-textline_0-segment_2-segCount3-cert"
    target="#f11-textblock_0-textline_0-segment_2-segCount3-text" locus="
    value" degree="0.9064"/>
<!-- ... -->
</zone>
```

(b) Le <zone> du <String> en TEI

FIGURE 8.6 – La modélisation du <String>

### 8.1.5 Le glyphe

Les caractères et de la ponctuation prédits par des modèles HTR sont tous encodés dans l'élément <Glyph> selon le schème ALTO. Contrairement au <String> qui sert à contenir une chaîne de glyphes, le <Glyph> du fichiers ALTO contient à la fois le masque et le texte. Pour cette raison, la modélisation en TEI représente, comme d'habitude, le masque du <Glyph> dans l'élément <zone> et le texte prédit dans l'élément <c>. Ce dernier est un élément du schème TEI destiné à la représentation d'un caractère, soit une lettre, soit de la ponctuation. Il convient bien donc à la représentation de toute prédiction dans l'élément <Glyph> du fichier ALTO. Le modèle HTR évalue son taux de réussite de sa prédiction du glyphe. L'évaluation de sa prédiction est représentée dans l'attribut @GC de l'élément <String>. L'acronyme *GC* signifie en anglais *glyph confidence*. L'exemple du <Glyph> et l'exemple de sa modélisation en TEI sont donnés dans la Figure 8.7. Une

visualisation de la transformation d'ALTO à TEI est donnée dans la Figure 8.12.

```
<Glyph ID="char_1" CONTENT="M" HPOS="837" VPOS="777" WIDTH="159" HEIGHT="
    162" WC="0.8127">
  <Shape>
    <Polygon POINT="..."/>
  </Shape>
<!-- ... -->
</String>
```

(a) Le <Glyph> en ALTO

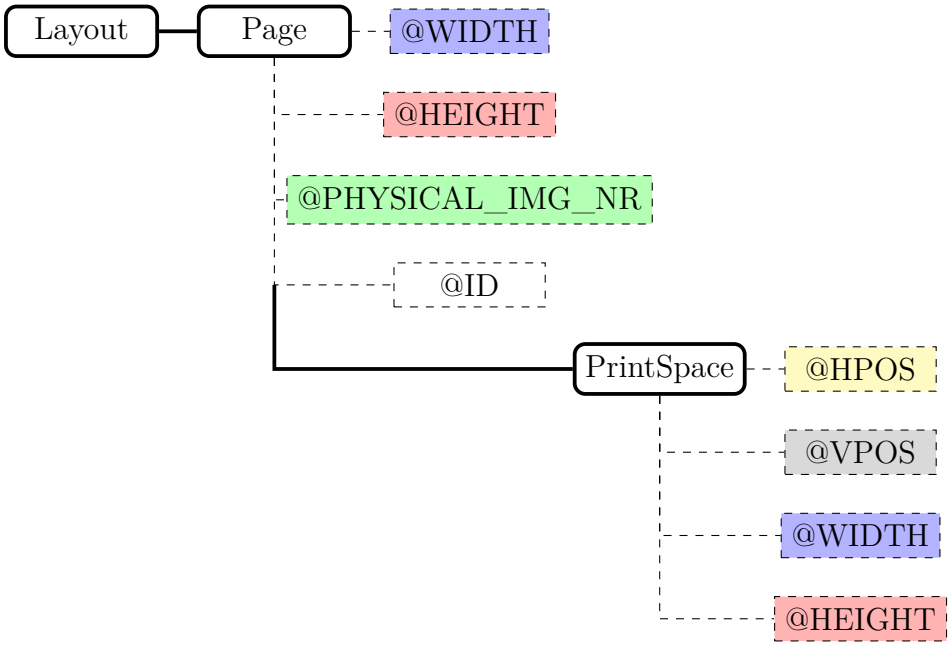
```
<zone xml:id="f11-textblock_0-textline_0-segment_2-char_1-glyphCount2" type
    ="String" ulx="837" uly="777" lrx="996" lry="939" points="..." source="
    https://gallica.bnf.fr/iiif/ark:/12148/btv1b8610802d/f11
    /837,777,154,120/full/0/native.jpg">
  <certainty xml:id="f11-textblock_0-textline_0-segment_2-char_1-
    glyphCount2-cert" target="#f11-textblock_0-textline_0-segment_2-char_1-
    glyphCount2-text" locus="value" degree="0.8127"/>
  <c xml:id="f11-textblock_0-textline_0-segment_2-char_1-glyphCount2-text">
    M</c>
</zone>
```

(b) Le <zone> du <Glyph> en TEI

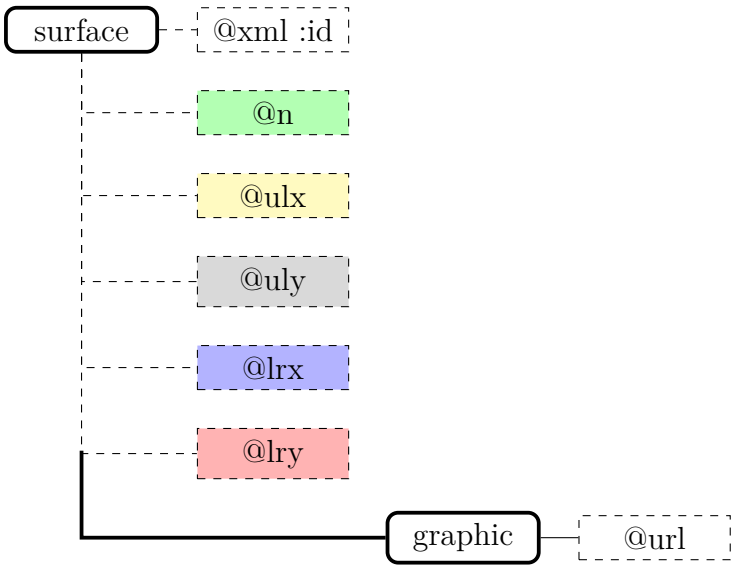
FIGURE 8.7 – La modélisation du <Glphy>

## 8.2 Les visualisations de la transformation

Des visualisations de la modélisation de chaque élément du fichier ALTO sont montrées dans les figures qui suivent. Les attributs sont visualisés par les carrés en ligne tirée et les éléments XML sont visualisés par les carrés en ligne solide. La couleur de l'élément ou de l'attribut du schème ALTO est répétée dans la visualisation du schème TEI quand sa valeur est utilisée.

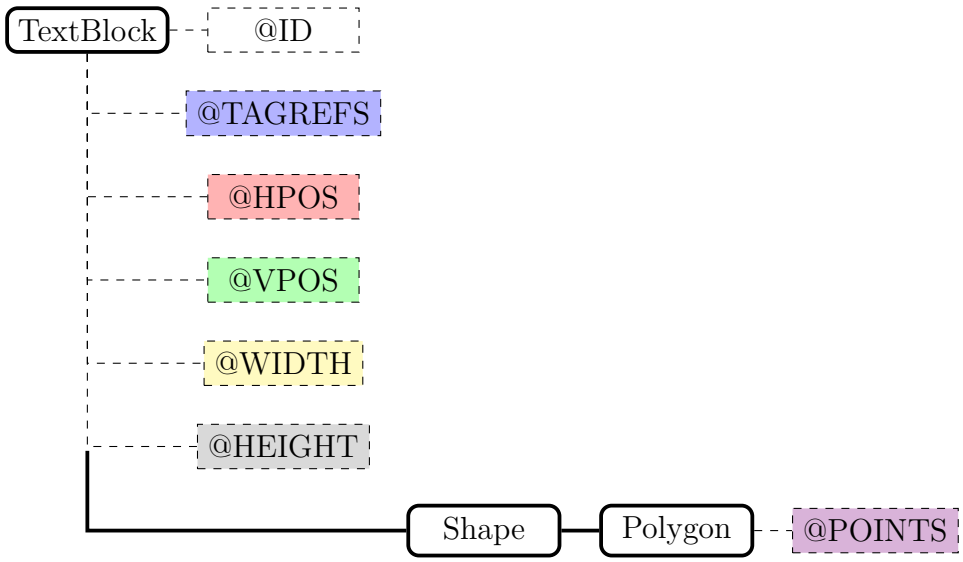



(a) Le modèle du <Page> en ALTO

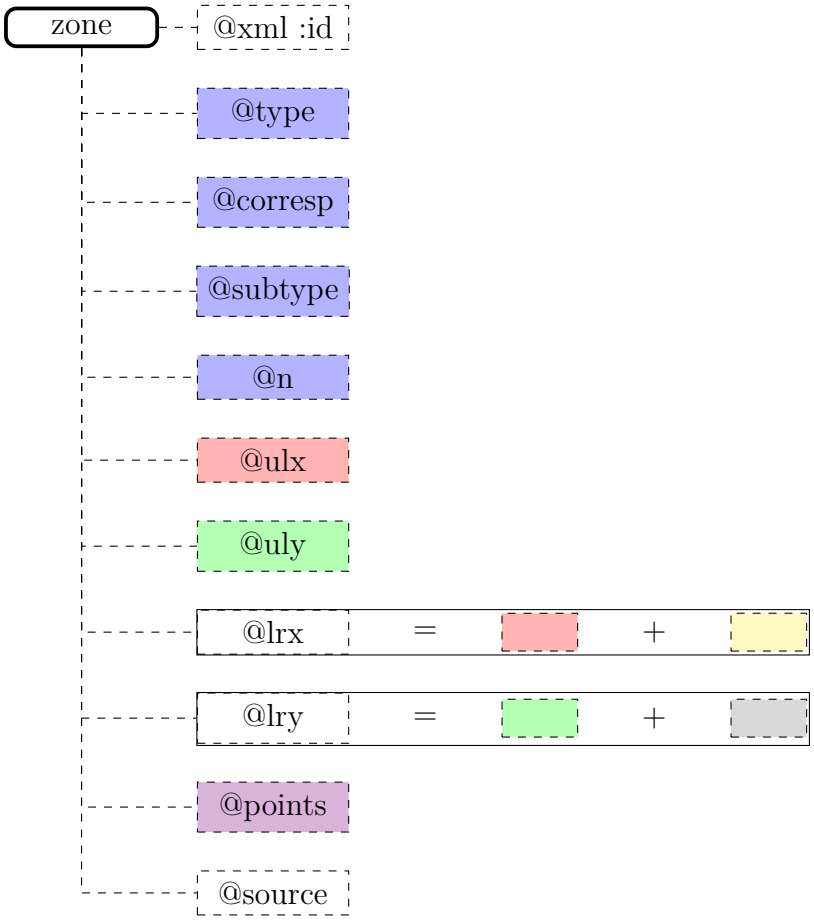



(b) La modélisation du <Page> en TEI

FIGURE 8.8 – La transformation du <Page> d'ALTO à TEI

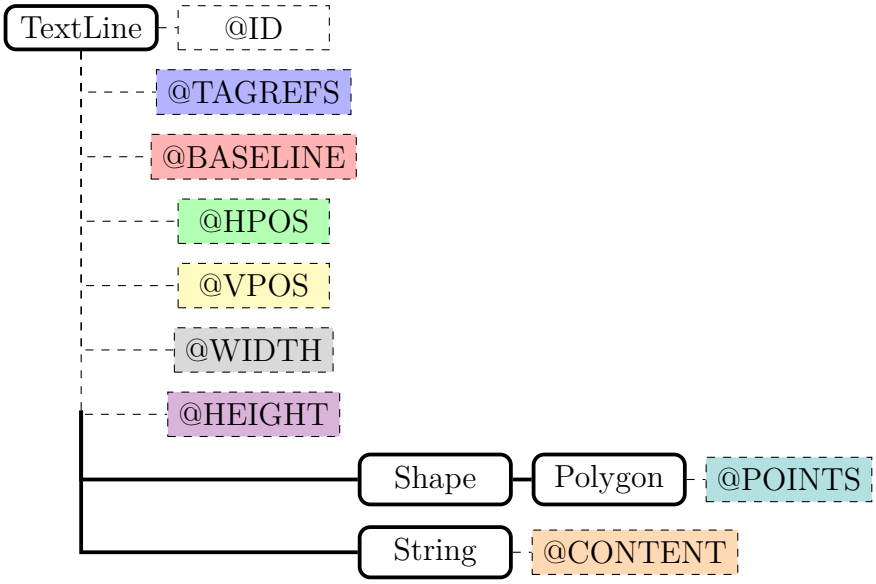



(a) Le `<TextBlock>` en ALTO

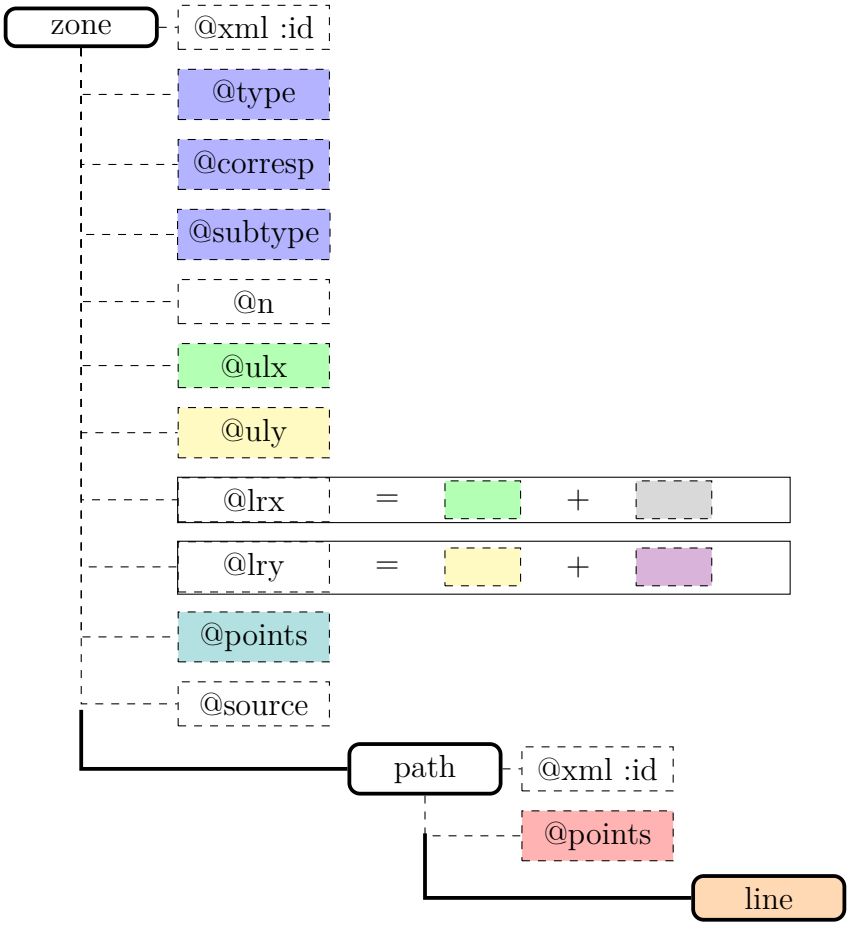



(b) La modélisation du `<TextBlock>` en TEI

FIGURE 8.9 – La transformation du `<TextBlock>` en TEI




(a) Le `<TextLine>` en ALTO où tout le contenu textuel prédit de la ligne est présenté dans l'attribut `@CONTENT`




(b) La modélisation du `<TextLine>` en TEI

FIGURE 8.10 – La transformation du `<TextLine>` en TEI

(a) Le `<String>` en ALTO

(b) La modélisation du `<String>` en TEI

FIGURE 8.11 – La transformation du `<String>` en TEI

(a) Le `<Glyph>` en ALTO

(b) La modélisation du `<Glyph>` en TEI

FIGURE 8.12 – La transformation du `<Glyph>` en TEI

# Chapitre 9

# Le traitement des données produites

Jusqu'à présent dans l'exposition sur la modélisation des fichiers ALTO en TEI, nous avons parlé de deux éléments descendant directement de la racine TEI. Le `<teiHeader>` renseigne sur les trois objets de texte concernés par la ressource numérique : la ressource elle-même, le fac-similé numérique dont les images des modèles HTR ont traité, et le document source physique à partir duquel le fac-similé a été fait. Le document TEI contient aussi des métadonnées utiles à l'exploitation de la ressource numérique. Dans un deuxième temps, le `<sourceDoc>` a récupéré toute donnée significative des fichiers ALTO et les met dans un arbre TEI, spécifiquement dans les éléments descendant du `<sourceDoc>` et des attributs. Ayant récupéré toutes les données des sources externes, l'application `alto2tei` traite ensuite les données déjà présentes dans la ressource en cours de construction.

La ressource numérique présente deux versions du texte prédit que l'application `alto2tei` produit à partir des données qu'elle a traités et mises dans le `<sourceDoc>`. Dans un premier temps, elle présente dans l'élément `<body>` une version du texte pré-éditorialisée, c'est-à-dire dans la manière par laquelle le texte s'apparaît sur la page. Les fautes d'orthographe, les sauts de ligne, les coupures de mot sont tous conservés dans la représentation du texte de l'élément `<body>`. La version pré-éditorialisée sert à l'analyse du texte transcrit ainsi qu'à l'analyse linguistique que peuvent vouloir faire des chercheurs.

Dans un deuxième temps, la ressource numérique du pipeline *Gallic(orpor)a* présente son propre analyse linguistique du texte transcrit grâce aux modèles TAL qu'elle met en pratique. La représentation du texte ainsi analysé, avec des entités nommés et des mots normalisés, est contenu dans l'élément TEI `<standOff>` pour qu'elle ne soit pas traité comme la transcription du texte. La transcription du texte est représentée dans l'élément `<body>` qui peut être facilement exploité par les éditeurs et les visionneurs de texte en format TEI. L'état actuel de l'application `alto2tei` n'effectue pas d'analyse linguistique, mais elle prépare la représentation du texte pré-éditorialisé du `<body>` duquel un *feature* ajouté plus tard pourrait disposer pour mettre en œuvre l'analyse linguistique produite par des modèles TAL.

## 9.1 La génération du `<body>` grâce au vocabulaire *SegmOnto*

L'objectif du `<body>` est de permettre aux logiciels d'exploiter la transcription du texte que des modèles HTR ont prédit. Cependant, des modèles prédisent plusieurs aspects de la page numérisée, plus que celui qui concerne le texte du document. Selon le vocabulaire *SegmOnto*, par exemple, les parties de l'image dans lesquelles s'encadre un dessin (*GraphicZone*), ou un numéro de page (*NumberingZone*), ou un titre en tête (*RunningTitleZone*) ne devraient pas être incluses dans la représentation du texte pré-éditorialisé. La plupart des chercheurs qui désireront analyser la transcription du fac-similé auront besoin du texte appartenant à l'œuvre telle qu'elle se conçoit. Les en-têtes, les numéros de pages, le texte des tampons, et les notes ajoutées après la publication de l'imprimé ou l'apparition du manuscrit sont tous intéressants à la recherche, mais ils dérangent l'analyse computationnelle de l'œuvre qui compte sur une représentation du texte continu, où les mots coupés au saut de ligne sont recollés et le texte qui continue sur la page suivante fait partie d'un encodage sans une telle interruption.

L'application `alto2tei` met en œuvre deux étapes pour extraire et nettoyer le texte transcrit. Dans un premier temps, elle recherche tous les éléments `<line>` du `<sourceDoc>` afin de recueillir les lignes de texte prédit dans le document source. Ces lignes deviennent directement de la prédiction des modèles HTR. Il faut donc les nettoyer. L'application garde toute saut de ligne à la fin de chaque chaîne qu'elle extrait du `<sourceDoc>` sauf si elle coupe un mot, ce que l'application sait si la ligne se termine par un tiret. Dans un tel cas, elle recolle le début du mot à la fin de ligne et le reste du mot au début de la ligne suivante.

Dans un deuxième temps, l'application `alto2tei` analyse les étiquettes de toutes ses lignes de texte ainsi nettoyées. L'équipe *Gallic(orpor)a* a déterminé une traduction entre les étiquettes du vocabulaire *SegmOnto* et l'arborescence de la TEI. La Table 9.1 montre quelles lignes font partie du texte principal et comment notre modélisation les balise dans des éléments du `<body>`. Toute ligne se précède par un `<lb/>`, l'élément du schème TEI destiné à indiquer le début d'une ligne de texte. Balisant le `<lb/>` est (au moins) l'un des éléments TEI de la Table 9.1.

| *SegmOntoZone* | *SegmOntoLine* | représentation en TEI |
|---|---|---|
| NumberingZone | DefaultLine | `<fw type="NumberingZone">` |
| QuireMarksZone | DefaultLine | `<fw type="QuireMarksZone">` |
| RunningTitleZone | DefaultLine | `<fw type="RunningTitleZone">` |
| MarginTextZone | DefaultLine | `<note type="MarginTextZone">` |
| MainZone | DefaultLine | `<ab type="MainZone">` |
| MainZone | DropCapitalLine | `<hi rend="DefaultLine">` |

FIGURE 9.1 – Les étiquettes du texte principal

Les lignes d'affilé dans une *MainZone* s'encadrent toutes dans l'élément `<ab>`, mais la ligne d'un *drop capital* se balise dans l'élément `<hi>` qui lui-même se balise dans le `<ab>` de la zone à laquelle la ligne appartient, comme se voit dans la Figure 9.2.

Ligne de texte
Deuxième ligne de texte

```
<ab corresp="#sourceDocBlockID" type="MainZone">
  <lb corresp="#sourceDocLine1ID"/><hi rend="DropCapitalLine">L</hi>igne de
    texte.
  <lb corresp="#sourceDocLine2ID"/>Deuxième ligne de texte.
</ab>
```

FIGURE 9.2 – Les lignes de la *MainZone*

Ainsi encodée, où les lignes de texte se suivent l'une après l'autre dans l'élément `<body>`, en ignorant les zones détectées qui ne portent pas de texte, la ressource numérique peut être exploitée par un logiciel configuré pour le TEI. Normalement l'élément `<lb/>` produira une saut de ligne pour que le texte qui traîne derrière l'élément vide s'apparaisse au début d'une nouvelle ligne. En plus, en recollant les mots coupés à la fin de ligne dans le document source, les mots ainsi nettoyés et reconstitués peuvent être cherchés grâce aux outils de recherche du logiciel. Par exemple, une utilisatrice ou un utilisateur ne trouverait pas le mot *bâtiment* s'il avait été écrit dans le document source comme *bâti-ment*, avec une saut de ligne après le tiret, et qu'il n'était pas recollé avant la recherche de l'utilisatrice ou l'utilisateur.

Enfin, le texte hiérarchisé du `<body>` peut supporter des annotations et de l'analyse encore plus profonde et particularisée au genre du document source. Par exemple, une équipe de chercheurs pourraient développer un système de filtrage, tel qu'une feuille de transformation XSL, pour traiter les répliques d'un drame que notre modélisation balise dans le `<ab>` d'une *MainZone*. Un tel exemple d'élaboration de l'encodage du `<body>` est visualisé dans la Figure 9.3. Le texte pré-éditorialisé du `<body>` permet d'élaborer un texte éditorialisé grâce à la base produite par l'application `alto2tei` et conceptualisée par l'équipe *Gallic(orpor)a*.

## 9.2 L'analyse linguistique

Le dernière aspect de la ressource numérique qu'avait conceptualisé l'équipe du projet *Gallic(orpor)a* est une analyse linguistique du texte prédit. Mon application `alto2tei` ne réalise pas encore l'application de cet objectif. Cependant, sa structure supporte l'ajout plus tard d'un *feature* qui soit reprendra le texte nettoyé du `<body>`, soit reprendra le texte que mon application a structuré dans les éléments du `<sourceDoc>`. Les deux options sont

```
<ab corresp="#sourceDocBlockID" type="MainZone">
  <lb corresp="#sourceDocLine1ID"/>Angélique.
  <lb corresp="#sourceDocLine2ID"/>Regarde-moi un peu.
  <lb corresp="#sourceDocLine3ID"/>Toinette.
  <lb corresp="#sourceDocLine4ID"/>Hé bien! je vous regarde.
</ab>
```

(a) Le texte pré-éditorialisé du `<body>`

```
<div2 type="scene" n="4">
  <sp>
    <speaker corresp="#sourceDocLine1ID"/>Angélique.</speaker>
    <p>
      <lb corresp="#sourceDocLine2ID"/>Regarde-moi un peu.
    </p>
  </sp>
  <sp>
    <speaker corresp="#sourceDocLine3ID"/>Toinette.</speaker>
    <p>
      <lb corresp="#sourceDocLine4ID"/>Hé bien! je vous regarde.
    </p>
  </sp>
</div2>
```

(b) Le texte éditorialisé

FIGURE 9.3 – La transformation des répliques prédits dans une *MainZone*

intéressants parce qu'elles permettent des approches différentes à la mise en œuvre des modèles TAL (Traitement automatique des langues). L'un des enjeux de l'analyse linguistique et l'alignement entre le texte saisie et le texte produit. Idéalement, un mot normalisé par des modèles TAL se lierait toujours à sa version transcrite depuis le document source. Ainsi, la donnée que des modèles TAL produite ferrait référence à l'identifiant du élément `<zone>` dans le `<sourceDoc>`.

Le défi à surmonter est que les modèles TAL ont besoin des mots complets, c'est-à-dire recollés s'ils était coupés à la fin de ligne. La transcription représentée dans le `<sourceDoc>` donne à chaque segment de texte son propre identifiant ; par conséquent, un mot peut porter deux identifiants s'il est coupé en deux. Il est possible de lier les deux identifiants du mot dans la transcription avec le seul identifiant du même mot nettoyé et donné aux modèles TAL. Mais la complexité qui se produit pour certains mots et pas pour des autres est un défi qu'une application doit pouvoir surmonter.

Un autre défi qui s'applique tous le temps concerne les lignes de texte de la transcription. Normalement les modèles TAL saisissent les segments du texte qui ont une cohérence lexicale, afin qu'ils puissent en chercher la signification linguistique de la phrase. Il faut donc leur donner des lignes de texte recollées. Comme l'application `alto2tei` cherche les tirets à la fin de ligne, afin de recoller les mots coupés, une application qui met en œuvre

l'analyse linguistique devrait chercher les signes de ponctuation qui indique le début ou la fin d'une phrase cohérente. Pour le français moderne, qui a déjà adopté des règles quant à la ponctuation et à l'emploi de lettres majuscules, un tel filtrage du texte est facile et déjà mis en pratique par plusieurs applications. Mais pour les documents historiques du moyen français le défi devient plus compliqué. En outre, il se complique encore quand on veut élaborer un système pour recoller les lignes de texte performant pour plusieurs états du français.

Dans le cadre du stage, j'ai expérimenté avec la mise en œuvre de l'analyse linguistique en saisissant le texte que mon application a préparé pour le `<body>` de la ressource numérique. Puisque les mots du `<body>` sont déjà recollés, la première étape est de composer des phrases qui ont d'une cohérence lexicale sur laquelle des modèles TAL peuvent s'appuyer. J'avais déjà écrit un code pour faire cela dans le cadre de la création des vérités de terrain. Il y avait une équipe qui s'occupait des vérités de terrain pour les modèles HTR et une autre qui s'occupait de la lemmatisation du texte transcrit par la première équipe. Ce dernier jeu des vérités de terrain est ciblé à entraîner des modèles TAL pour l'application de l'analyse linguistique au corpus. Mon code a aidé l'équipe qui s'en chargeait car il traite les lignes de texte et sort un fichier TXT. Le texte se donne soit dans des phrases complets, soit dans une phrase partielle qui ne termine pas par des signes de ponctuation en fin de phrase, tel comme le point, le point d'exclamation, et le point d'interrogation.

Normalement, des modèles TAL saisissent des listes itératives des phrases et des mots [1]. Ayant préparé la liste itérative des phrases d'une cohérence lexicale, grâce à mon script, l'expérience a profité des scripts développés dans le cadre du projet *e-Ditiones* par Alexandre Bartz [2]. L'une des premières étapes de l'analyse linguistique, et quelque chose que fait le script de Bartz, est la tokenisation, par laquelle la phrase est désagrégée en *tokens*, c'est-à-dire des petits unités lexicale. Le texte ainsi atomisé peut être ensuite traité par des modèles TAL entraînés pour faire certaines tâches, telle comme la lemmatisation et la normalisation.

Le script de Bartz apport la lemmatisation au texte grâce à la librairie pytho *pie-extended*, développée par Thibault Clérice [3]. Un modèle entraîné pour faire la lemmatisation trait tout *token*, que le script de Bartz a préparé, et renvoie deux genres de données : la forme canonique du *token* (le *lemme*) et une suite d'analyses en fonction de la nature du

1. Pedro Javier Ortiz Suárez, Benoît Sagot et Laurent Romary. « Asynchronous Pipeline for Processing Huge Corpora on Medium to Low Resource Infrastructures ». In : *7th Workshop on the Challenges in the Management of Large Corpora (CMLC-7)*. Sous la dir. de Piotr Bański et al. Cardiff, United Kingdom : Leibniz-Institut für Deutsche Sprache, juill. 2019. doi : 10.14618/IDS-PUB-9021. url : https://hal.inria.fr/hal-02148693 (visité le 07/09/2022).

2. Alexandre Bartz et Juliette Janes. *Annotator*. E-ditiones. url : https://github.com/e-ditiones/Annotator (visité le 07/09/2022).

3. Thibault Clérice. *Pie Extended, an Extension for Pie with Pre-Processing and Post-Processing*. Zenodo, juin 2020. doi : 10.5281/zenodo.6534764. url : https://zenodo.org/record/6534764 (visité le 07/09/2022).

*Humble salut et recognoissance de sa liberalité enuers luy.*

(a) Le texte de saisie en phrase cohérente

| *token* | *lemma* | *pos* |
|---|---|---|
| Humble | Humble | ADJqua |
| salut | salut | NOMcom |
| et | et | CONcoo |
| recognoissance | recognoissance | ADJqua |
| de | de | PRE |
| sa | son | DETpos |
| liberalité | liberalité | NOMcom |
| enuers | enuer | VERcjg |
| luy | luy | PROind |
| . | . | PONfrt |

(b) La tokenisation et lemmatisation en format CSV

| *token* | *lemma* | normalisation |
|---|---|---|
| Humble | Humble | Humble |
| salut | salut | salut |
| et | et | et |
| recognoissance | recognoissance | reconnaissance |
| de | de | de |
| sa | son | sa |
| liberalité | liberalité | liberalité |
| enuers | enuer | envers |
| luy | luy | lui |
| . | . | . |

(c) La normalisation en format CSV

FIGURE 9.4 – L'expérience du Traitement automatique des langues sur le document d'ARK `bpt6k724151`

*lemme*[4]. Après la lemmatisation, le script de Bartz applique un modèle de normalisation qui transforme le français du document source en une version moderne. Dans l'expérience que j'ai faite avec un imprimé du XVIe siècle (ARK bpt6k724151), j'ai utilisé un modèle de normalisation entraîné par Rachel Bawden[5]. Un exemple des résultats des deux tâches de TAL est montré dans la Figure 9.4.

Le dernier défi pour une application éventuelle de l'analyse linguistique dans le pipeline *Gallic(orpor)a* est la modélisation des données produites. La sortie des modèles TAL est typiquement du format CSV (Comma Separated Values) ou TSV (Tab Separated Values), mais la ressource numérique exige des données en XML et formatées selon les

4. Dans l'analyse du script de Bartz, la nature (*part of speech* ou POS) du *lemme* est accompagnée par des analyses supplémentaires et dépendents sur la langue. Par exemple, il donne au *lemme* identifié comme un verbe son temps, et celui identifié comme un nom son genre.

5. Rachel BAWDEN. *ModFr-Normalisation*. URL : `https://github.com/rbawden/ModFr-Norm` (visité le 07/09/2022).

normes de la TEI. Une telle modélisation a été présentée par Bartz, Juliette Janes, Laurent Romary, Philippe Gambette, Rachel Bawden, Pedro Ortiz Suaréz, Benoît Sagot, et Simon Gabay en 2021 à la conférence TEI[6]. L'équipe *Gallic(orpor)a* envisage à élaborer une modélisation du `<standOff>` similaire. Ainsi disposé du script de l'application `alto2tei` qui prépare une version pré-éditorialisée du texte, le pipeline *Gallic(orpor)a* est prêt à avoir intégrée de l'analyse linguistique et pouvoir construire le `<standOff>`.

6. Alexandre Bartz et al. « Expanding the Content Model of annotationBlock ». In : *Next Gen TEI, 2021 - TEI Conference and Members' Meeting*. Virtual, United States, oct. 2021. URL : `https://hal.archives-ouvertes.fr/hal-03380805` (visité le 07/09/2022).

# Conclusion

# Annexe A

# Données

## A.1 Portée des données d'entraînement

| Type | Genre | Forme | Écriture | Siècle | Langue | Titre |
|---|---|---|---|---|---|---|
| manuscrit | poésie | vers | gothique | 13 | fro | Français 20050 - chansonnier de Saint-Germain-des-Près |
| manuscrit | récit | prose | gothique | 13 | fro | Français 23117, légendier |
| manuscrit | récit | prose | gothique | 13 | fro | Français 6447, légendier |
| manuscrit | récit | prose | gothique | 13 | fro | NAF 23686, légendier |
| manuscrit | récit | prose | gothique | 13 | fro | Français 13496, légendier |
| manuscrit | récit | vers | gothique | 13 | fro | Français 860 - Roland, Gaydon, Ami et Amile, Jourdain de Blaye, Auberi le Bourguignon |
| manuscrit | récit | vers | gothique | 13 | fro | Français 12615 - chansonnier de Noailles |
| manuscrit | récit | vers | gothique | 13 | fro | Français 1443 - Garin le Lohérain (C) et Girbert de Metz |
| | | | | | | Français 12603 - Fierabras, mais aussi Chevalier as deux espees |

# Table des figures

# Liste des tableaux

# Table des matières