An accessible and transparent pipeline for publishing historical egodocuments

Alix Chagué, Floriane Chiffoleau

▶ **To cite this version:**

Alix Chagué, Floriane Chiffoleau. An accessible and transparent pipeline for publishing historical egodocuments. 2021. hal-03180669

HAL Id: hal-03180669

https://hal.archives-ouvertes.fr/hal-03180669

Preprint submitted on 25 Mar 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution| 4.0 International License

An accessible and transparent pipeline for publishing historical egodocuments

Alix Chagué[1] and Floriane Chiffoleau[1]

[1]ALMAnaCH, Inria, Paris, France, name.surname@inria.fr

March 17, 2021

Presentation script

Why a pipeline ?

When working on the transcription and edition of a corpus of archival documents, we often find ourselves in two types of situations:

- Either we're running a blackbox that does almost everything we want, but gives us no power to intercept data anywhere in the pipeline except at the end, and doesn't let us customize a specific step that would be of importance to the project.
- Or we're led to use what we can call a scattered toolbox, made of various software, which exposes us to several risks:
  - the obsolescence of a single one of these tools would question the integrity of the whole scenario;
  - the portability of data from one software to the next is not guaranteed and might force us either to waste time or to sacrifice a portion of our (meta)data.

Finding a middle ground between these two extreme scenarios is key to ensure an efficient processing of a corpus where the integrity of the data is guaranteed and where the user has complete power over every step of the pipeline. So within the DAHN project, we decided to develop and document a turnkey pipeline for scientific edition which meets these stakes.

The DAHN project

In 2020, Inria, the EHESS and the University of Le Mans created a scientific and technological partnership, with the support of the French Ministry of Higher Education, Research and Innovation (MESRI), under the project named DAHN (Dispositif de soutien à l'Archivistique et aux Humanités Numériques)[1].

Its goal is to facilitate the digitization of data extracted from archival collections and their dissemination to the public in the form of digital documents in various formats and/or as an online edition.

One of the main productions of this project is indeed the documentation of a complete scenario to automatically create a scientific digital edition starting from a corpus of physical documents. As a way to emphasize the importance of intermediary products and modularity, attention is paid to the reusability of the data at any point in the scenario.

[1]The project has a repository: https://github.com/FloChiff/DAHNProject

A most remarkable trait of the project is also its interdisciplinarity. By mixing these three institutions, the group is made of engineers and researchers, who provide different points of view on the same project, definitely rooting it in the field of Digital Humanities.

The mission of the ALMAnaCH team is to choose, develop and enhance the tools that will be part of the pipeline created for the transformation of the corpus. We have to think about these tools in terms of adequation to the user's goals, ergonomics, accessibility, and, obviously, the portability of formats.

The corpus I: Paul d'Estournelles de Constant correspondence

To elaborate and test our pipeline, we use a study corpus made of a correspondence between two diplomats dating from the First World War period, and continuing a few years after. During this period, the French senator, Paul d'Estournelles de Constant, wrote to his American friend and colleague, Nicholas Murray Butler and exposed the French situation as well as his defense for Peace. One of the major points that determined our choice is the duality in this corpus: these letters are from the private sphere, they're a personal correspondence where d'Estournelles shares his experience and his opinion. But they bear a strong public dimension, because they were intended, beyond their initial recipient, to be published in American newspapers as a report of the war and a motivation for the United States to engage in it.

This corpus is of genuine historical interest but it is also a perfect test subject because of its size and content. The correspondence is made of about 1 500 letters, 430 from the war period alone (from April 15, 1914 to November 19, 1918), with lengths varying from one page to several tens which means that it is a big enough corpus for automatic transcription to be relevant.[2]

The corpus II: Digitization of the corpus

The first step of our pipeline and as a matter of fact, of every digital processing of an ensemble of documents, is its digitization. Our corpus is peculiar because its digitization is not homogenous. We started our experiments with a set of photographs taken for another use by a member of the project prior to its creation. It sometimes results in cropped images, blurred lines and incomplete pages which can slow or complicate the process, but it also shows that for the most part, the pipeline does not require high standards images to function.

We are currently in discussion with the Archives Départementales de la Sarthe, the institution where the collection is kept, to start an institutional and exhaustive digitization campaign with proper material. The goal here is to be provided with high quality images for the subsequent publication. Once we have those images and their data, we will import them into a IIIF server, in order to have them available for our web application.

The pipeline

The pipeline we are developing aims at going from point A (the document is digitized) to point B (the document is published). To do so, there is a few steps that need to be executed:

- digitization,
- segmentation,
- transcription,
- post-process correction,
- encoding ; and
- publication.

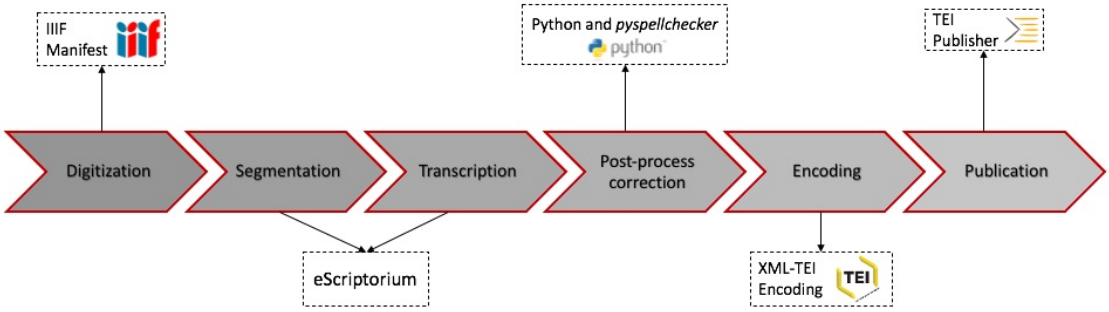

Figure 1: Pipeline for the scientific digital edition of a corpus

In our scenario, we want to execute these steps one after the other without encountering problems, such as the ones we could have with a blackbox or a scattered toolbox.

The core software in our pipeline is eScriptorium, but we also rely on several tools and frameworks such as IIIF for the distribution of images, TEI XML for the encoding of the transcriptions and TEI Publisher for their publication. For now, Python programs - which is the language used to develop the eScriptorium application- are used at different points to serve the data from one step to the next.

The OCR engine: a capstone

The capstone in our scenario is thus the use of an OCR engine to produce a transcription. Various software are available, but we decided to base our pipeline on the eScriptorium and Kraken solution.

eScriptorium is a web interface for collaborative and automatic transcription projects: it enables the user to load sets of images, to associate them with metadata, to perform segmentation and layout analysis, to perform transcription, to train models for these two steps ; and to export the resulting data in various formats as well as to export the models. The platform currently uses Kraken as a core OCR engine but it is intended to be an agnostic shell for any OCR engine if one wishes to use another solution.

There are several reasons for our choice:

- there are historical ties between the ALMAnaCH team and SCRIPTA (the EPHE project team behind eScriptorium);
- the platform gives the user a lot of freedom in terms of sharing data, documents and models and its interface is rather intuitive;

and most importantly:

- it uses up-to-date import and export standards such as ALTO XML and PAGE XML;
- it is entirely open-source, so not only are we sure never to be stuck with a blackbox, but we're also able to build adds-on for the platform to gather into one place every tool we need.

Overall we've designed a pipeline relying exclusively on open-source services and software and we intend it to comply with the FAIR principles[3].

[2]Schematic representation of the corpus are available in the annexes with the figures 2 and 3.
[3]Findability, Accessibility, Interoperability and Reusability

The pipeline step by step

Segmentation and Layout Analysis

The segmentation and annotation of the layout is taken on by a system of line and region tagging coupled with an ontology. It is therefore possible to train with Kraken a segmentation model that automatically tags the types of zones and lines according to a modelization of the layout which fits exactly the needs of a specific project. Of course it is possible to rely on eScriptorium's default segmentation model as well.

It is valuable to be able to build our own modelization and pass it on to a segmentation model because it makes possible the integration of a framework such as the one provided by TEI standards as early as in the lay-out analysis step, which then facilitates the elaboration of a TEI output down the pipeline.

On this note, the DAHN project participates in the SegmOnto[4] initiative which is a working group aiming at creating a general TEI-based ontology for HTR projects, within eScriptorium.

Transcription strategy

Then, to generate our transcription model, we coupled ground truth based on our own corpus with a generic model for French typewritten text trained on a corpus of ground truth[5] which we created besides[6].

Let's briefly go over a few key rules we applied while creating the ground truth for our transcription model. Our documents contain mostly typewritten text, but they also include handwritten insertions and manual strokes. For these manually added components, we trained our model not to transcribe them but merely to signal them with a symbol (£ or €). Then handwritten insertions are transcribed manually (because there are very few) in a post-processing step.

Post-process corrections[7]

The next step is the post-process corrections, which are essential to finish the transcription and have it completely ready for encoding. This phase includes several tasks: some of which are automatically performed, the rest manually (nb: we already mentioned the additions of handwritten insertions).

We use Python programs which include tools and libraries such as *pyspellchecker*[8], which as suggested by its name, spell-checks a text and suggests corrections based on the use of the Levenshtein distance and a dictionary. The program processes the words in the text, and automatically replaces the misspelled ones with their correction. The corrected text is then reimported into eScriptorium.

At the moment, these Python programs are peripheral to eScriptorium but they can be fully integrated within the platform's workflow thanks to its API.

Encoding

We have now fully transcribed texts, in TEXT format after exportation from eScriptorium, so we can proceed to the encoding. Our corpus is made of egodocuments, which are, to put it simply, documents written in first person reporting personal events. In a case like our corpus of letters, there is a pattern in the layout and the logical structure of the documents which can normally easily be rendered thanks to the TEI framework.

The header is not really specific to egodocuments. It encodes texts as usual, with the addition of a manuscript description to detail the information inherent to the corpus. The only specificity

[4] SegmOnto organization: https://github.com/SegmOnto

[5] Tapuscorpus dataset: `https://github.com/HTR-United/tapuscorpus`

[6] A first model trained from scratch on 84 pages gave unsatisfying results. We partially transcribed 100 more pages focusing on building a train set more representative of the whole corpus. This second model, trained from scratch as well, reached an accuracy of 92.74 %. Lastly, fine-tuning a generic model such as the Tapuscorpus one enabled us to reach an accuracy of 93.90 %. While these accuracy levels suggest that fine-tuning is not more efficient than training from scratch when working with typewritten text, we believe our third model is likely to give more constant results and to show a better robustness to image/text variations.

[7] A schematic representation of the steps is available in the annexes with the figure 4

[8] https://pyspellchecker.readthedocs.io/en/latest/

is in the *profileDesc*, which contains few tags dedicated to correspondence, that are specified and explained in the TEI Guidelines.

For most of the body, we use default text structure. The text is consequently encoded by using tags such as titles and paragraphs, as well as tags to encode the changes made in the text (*addition* and *deletion*), the difficulties presented by the corpus (*unclear*, *gaps*) and the named entities (*person*, *place*, *organization*). Then, we have to encode the correspondence part and that's where it can become more difficult. There are easy tags such as *opener* and *closer* and the tags nested in it like *address*, *signature* or *postscript* but there also are more complicated parts that require choices from us, like letterhead. To develop this XML tree, we used information from the TEI Guidelines[9], the TEI mailing-list[10] and the Special Interest Group dedicated to Correspondence[11] and we wrote an ODD for egodocuments, documenting all the choices we made.

Furthermore, to help with the encoding, we develop Python scripts to encode the metadata using an inventory filled with specific information and the body using regular expressions, helped by the pattern in the layout.

Publication

At this point in the pipeline, we possess images and TEI XML files ready to be published.

The final step consists of importing these TEI files in a specific application developed with TEI Publisher. TEI Publisher is an open-source application based on exist-db, which loads TEI XML files and generates a complete web application using transformations files and templates, to populate web pages. It's a very convenient way to quickly deploy an online edition of a corpus once the TEI edition is ready. Furthermore, it is easily customizable, and extensively documented.

Favoring the use of the pipeline: Documentation

Speaking of, one of the ways to ensure the transparency of a workflow is obviously to document it extensively. This way any person wishing to adapt it to their own corpus and objectives, can easily do so. The presence of documentation was one of our main concerns when developing the pipeline. There is on the one hand the initial documentation provided with the software, standards and framework we use, and then there is the documentation we produce ourselves. As we said before, we developed an ODD in TEI XML and wrote a comprehensive series of guidelines. All of our Python scripts currently existing outside of eScriptorium are properly commented and we publish them in the form of interactive Jupyter notebooks where it is possible to replicate a specific step of transformation. Finally, the data we produce is published on a dedicated Github repository where one can see what the data looks like at any point in the pipeline. It is therefore possible to replicate the entire pipeline from the beginning.

Generalisability & modularity

An experimentation we did with another corpus, the Berlin intellectuals[12], which is a corpus made of a correspondence from the start of the nineteenth century, enabled us to confirm two key aspects of our scenario: its modularity and its capacity to be generalized:

- It is possible to plug data in down the pipeline without following the steps from the beginning: as long as the corpus complies with the encoding rules we established, it is not necessary to have generated the TEI XML with eScriptorium and our scripts, to be able to publish them on our web application.
- Our encoding rules and the publication are not limited by space and time: the Berlin Intellectuals corpus and d'Estournelles' letters span over more than a hundred years and were written in two different languages.

[9] TEI Guidelines: https://tei-c.org/release/doc/tei-p5-doc/en/html/index.html
[10] TEI mailing-list: https://tei-c.org/support/
[11] *Correspondence* SIG: https://tei-c.org/Activities/SIG/Correspondence/
[12] Berlin intellectuals' website: https://www.berliner-intellektuelle.eu/?en

Furthermore, when addressing the question of sustainability, we wish to point to the fact that, on top of using standards widely adopted by the community, we ensure the dissemination of the data in useful formats. This does not simply relate to TEI XML files but also to the ground truth created to generate our transcription models. We contribute to the HTR-United project[13] which is a Commons project for HTR Ground Truth.

Key takeaways

So in terms of what we've gone through during this presentation, here are the key takeaways we would like to reiterate:

- in order to avoid a blackbox or scattered toolbox scenario, we rely on a scenario fully documented and based on open-source software and widely adopted standards;
- we build on the adaptability of the input and output formats of software like eScriptorium and TEI Publisher to go beyond our case scenario and consider the importance of modularity and the capacity of the scenario and tools to be generalized;

By way of conclusion, if you wish to use our scenario, all you need is:

- digitized images;
- a transcription system such as eScriptorium/Kraken;
- a TEI modelisation of the structure of your documents;
- and a server to deploy a TEI Publisher application.

[13] HTR-United organization: https://github.com/HTR-United/

Annexes

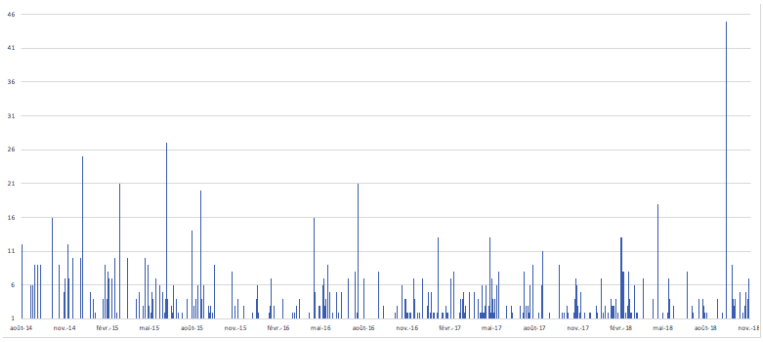

Figure 2: Number of pages per letters through time

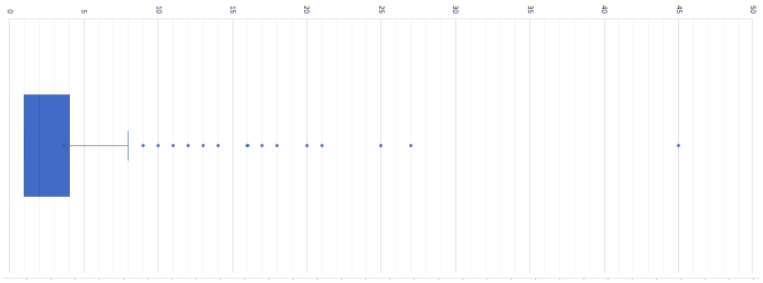

Figure 3: Distribution of pages per letter

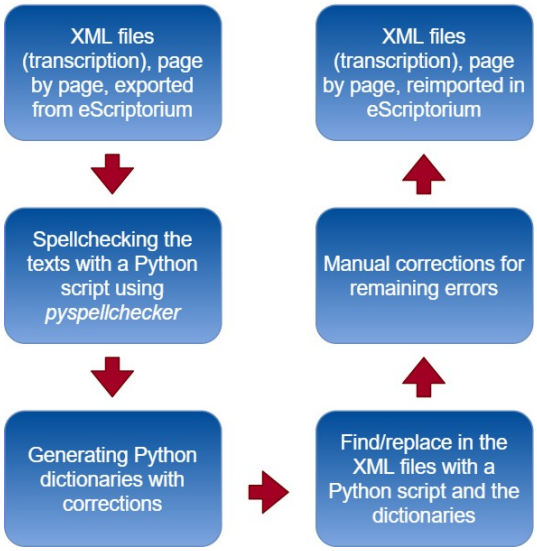

Figure 4: Steps for the post-process corrections