

From Manuscript to Metadata: experiments on Handwritten Text Recognition, Tagging and Importation for the *Memoriali* series (1265-1452)*

Edward Loss^(a)
Fabiana Guernaccini^(b)
Manuel Carassai^(c)

a) University of Genoa, <https://orcid.org/0000-0002-9837-8321>

b) Regesta.exe, <https://orcid.org/0000-0002-2101-882X>

c) Regesta.exe

Contact: Edward Loss, edwarddettmamloss@gmail.com; Fabiana Guernaccini, fguernaccini@regesta.com;
Manuel Carassai, mcarassai@regesta.com

Received: 30 December 2024; **Accepted:** 21 March 2025; **First Published:** 15 May 2025

ABSTRACT

The paper describes the experiment of creating a tailored tagging system for the *Memoriali* – a series of notarial records produced in Bologna almost uninterruptedly between the second half of the 13th and first half of the 15th century – using the Handwritten Text Recognition (HTR) software Transkribus. Product of the conjoint effort of the MemoBo project and Regesta.exe, the aim of this system is to facilitate the export of a specific set of information from these documents directly to a customised database on xDams. This tagging system explores a secondary feature of Transkribus, highlighting its potential and its limits for establishing a coherent workflow, which initiates with the application of *ad hoc* models for creating semi-automatic transcriptions of these late medieval notarial records and concludes with the exportation and proper placement of their data inside our xDams database.

KEYWORDS

HTR; Transkribus; Memoriali; xDams; Bologna.

* This work was developed as part of the European Union initiative – Next Generation EU, mission 4, component 1, ON: *Objects in network. The social life of things in the fifteenth century between notarial sources and semantic web*, PRIN2022 (P. I. Tommaso Duranti), CUP:J53D23000510006; Code MUR: 2022XTSEZ3_001. Edward Loss is the author of sections 1, 2, 4 and 5; Fabiana Guernaccini and Manuel Carassai are the authors of sections 3 and 6. All three authors wrote section 7.

1. Introduction

This paper describes the issues and challenges of creating a tailored tagging system for the *Memoriali* – a series of notarial records produced in Bologna almost uninterruptedly between the second half of the 13th and first half of the 15th century – using Transkribus and aimed at facilitating the export of a specific set of information from these documents directly to a customised database on xDams. Product of the conjoint effort of the MemoBo project and Regesta.exe, this tagging system explores a secondary feature of the Handwritten Text Recognition (HTR) software, highlighting its potential and its limits for establishing a coherent workflow, which initiates with the application of *ad hoc* models for creating semi-automatic transcriptions of the late medieval notarial records just mentioned and concludes with the exportation and proper placement of their data inside our xDams database. The article is thus divided into four main sections: the first one concerns the *Memoriali* project, its objectives and its custom-made database, produced by Regesta.exe. The categories of the database predate the project's entire experimentation with HTR, imposing, therefore, the key-elements of the tagging system; a peculiarity which marked some of its features. For this specific reason, this section includes a detailed and exhaustive description of the database.

The second section focuses on the experimentation of the MemoBo project with HTR, and more specifically, with Transkribus. It briefly describes a tortuous process, which culminated with the creation of Large HTR Models based on *Ground Truth*¹ (Hodel 2022) on the scale of hundreds of thousands of words, capable of not only successfully transcribing the late medieval Bolognese *Memoriali*, but also demonstrating potential for geographical transposition in the same chronological context. The section highlights how the projects' initial goal of producing semi-automatic transcriptions in order to aid the manual population of the database changed considerably once we realised Transkribus' tagging feature left an imprint in the XML version of the transcribed material, which could be potentially used by Regesta.exe to export this material directly into the xDams database.

Sections three and four constitute the core of the paper. They illustrate, on one hand, the different levels and elements of the customised tagging system, justifying how some of the decisions made were due to structural limitations of Transkribus. On the other hand, they specify all the key issues of mapping and converting a Transkribus XML tagged with this custom system into an XML file compatible with a xDams XML file for proper semi-automatic exportation.

The paper concludes with the complete structure of this tagging system, offering some general considerations on how its implementation constitutes a time-saving tool in comparison with manually cataloguing the *Memoriali*.

¹ In the context of Machine Learning and of Handwritten Text Recognition, the term *Ground Truth* means the portion of the material (transcribed text) provided to the software which will serve as the reference for any of its autonomous attempts to transcribe scanned text. Also known as *Golden Rule*, HTR software consider *Ground Truth* material as its source of learning, since these systems are not taking input from the open web. *Ground Truth* can either be generated completely manually by the user or by applying a previously existent model (such as “public models”, those made available by pre-existent research groups) and correcting the resulting transcriptions.

2. The MemoBo project

Created as a partnership between the University of Bologna, the State Archive of Bologna and Regesta.exe in 2020, the MemoBo project constitutes a big data initiative centred on a rich notarial late medieval series – the *Memoriali* – of nearly three million records produced in Bologna between 1265 and 1452 (Bruno et al. 2023). The scope of the *Ufficio dei Memoriali* – the institution from which the series and its surviving records borrow their name – was to safekeep the authenticity of notarial acts created inside the city’s jurisdiction. A statutory disposition established its existence in 1265 declaring that any notarial act valued on at least 20 Bolognese *lire* that was not properly brought to the office’s attention should not be considered valid (Continelli 1988, IX-XL). This legal obligation is at the root of the extraordinary value of the series for the study of the economic and social life of a large late medieval commune, which constitutes an almost unique case of a documentary series of this sort preserved almost uninterruptedly for nearly 200 years. The aim of the MemoBo project is to render available the information contained in the 322 tomes that hold these almost three million acts, transforming it into searchable, shareable and reusable data for scholars and the general public. Only a set of fundamental information is extracted from these records – at this stage the project’s goal is not to provide complete transcriptions of these sources – and the core of MemoBo is the custom-made database produced in xDams to host this data.

3. The database in xDams

As mentioned in the introduction, the experimentation with Transkribus constituted a posterior strategy of the initial objectives of the MemoBo project. Therefore, the entire experimentation and mapping operations were executed in the already implemented database. A parallel and temporary database was created to host the test data, considering a possible subsequent transfer of the material to the main database.

MemoBo’s xDams database adopts a hierarchical structure, allowing for the organization of levels for the description of volumes, registers, deeds, and finally, properties – movable, immovable, and rights - which are the subject of the deeds contained in the *Memoriali*.

The descriptive entry for “volume” and “register”

The databases’ initial macro-level entries are “volume” and “register”. These two allow the inclusion of identifying information such as:

- denomination
- registering notary
- chronology
- notes

The descriptive entry for “deed”

Under those two more general entries, one finds the entry “deed”, which allows the insertion of information regarding the individual notarial record, both from a physical and content perspec-

tive. This entry, thus, contains information such as *folio* identification, the position of the deed inside a specific *folio*, and the language of the document. These details trace the physical perimeter of the object, but also contain some extra information: in addition to the drafting and recording notary, the entry collects data on the topical date, the record's registration date, and indexes all of the people who are mentioned in the deed, with their respective roles. For each person, it is possible to insert details such as gender, name, surname, age, origin or residence, qualification, belonging or reference entity, condition, profession, function inside a specific deed, and any other eventual note. Each deed may have as its object one or more assets or properties, described in a specific sheet to maintain the peculiarities of each type: “movable properties”, “immovable properties” and “rights”.

The descriptive entry for “property”

The entry allows the description of the three types just mentioned, after selecting the type via a drop-down menu.

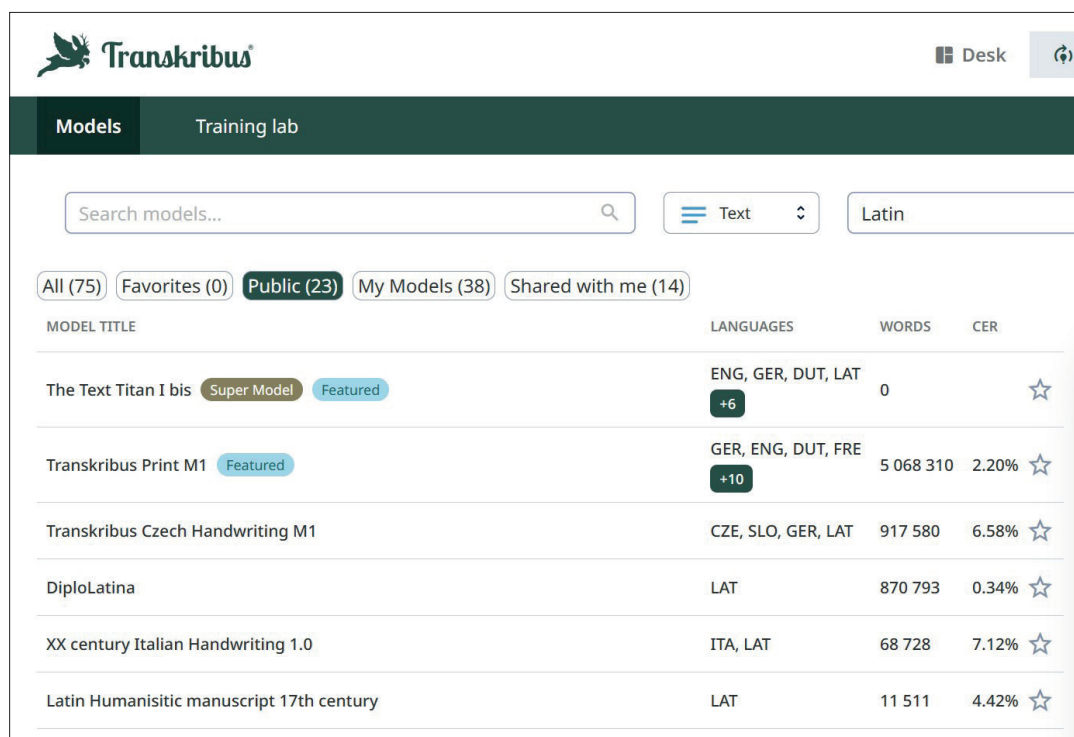
Figure 1. The first part of the descriptive entry for property on xDams.

The image provides an example of the creation of a property record. In the depicted case, the property is categorized as “rights”, but it is equally possible to create records for “movable property” or “immovable property”. The input form is organised into sections, filled out differently depending on the type of object previously selected. Subsequently, it is possible to specify the quantities, measurements, value, rent, and an index of the names of the people and places associated with the individual property. Together all these different categories of information constitute a set of over 200 potential elements that could be extracted from each deed of the *Memoriali* series.

4. MemoBo and HTR

The colossal dimension of the series together with the rather complex and detailed set of information selected for transposition into the database motivated the MemoBo project to look for digital solutions that could complement and overcome the time constraints imposed on manual cataloguers. In 2022, the project started experimenting with Handwritten Text Recognition software, especially with Transkribus, thanks to its popularity in the sector and its relatively user-friendly interface, in particular to scholars who have no background in computer sciences. The initial aim of the experiment was to produce fairly readable transcriptions with the software that could expand the pool of cataloguers recruited for the project – with transcriptions, no longer would we have to rely only on cataloguers highly skilled in Palaeography and late medieval Latin to populate the database.

The experimentation was divided into three phases: the first one focused on the creation of *pure models*, based on a single notary – a “single hand” – and was especially centred on the first volumes of the collection from the late thirteenth-century, which possesses a clearer handwriting from a palaeographical perspective. The idea was to use these models as *base-models* for more daring attempts to create *mixed-models* and *mega-models* able to cover bigger chronological portions of the collection. This initial approach was mostly motivated by the fact that there are no *public models* available at the platform that focuses both on late medieval Latin and on documents produced in the peninsula. As a matter of fact, the timeframe from the second half of the thirteenth- all the way into the first decades of the fourteenth-century is largely uncovered by the research groups that decided to make their models available to others in the open platform.



MODEL TITLE	LANGUAGES	WORDS	CER
The Text Titan I bis Super Model Featured	ENG, GER, DUT, LAT +6	0	☆
Transkribus Print M1 Featured	GER, ENG, DUT, FRE +10	5 068 310	2.20% ☆
Transkribus Czech Handwriting M1	CZE, SLO, GER, LAT	917 580	6.58% ☆
DiploLatina	LAT	870 793	0.34% ☆
XX century Italian Handwriting 1.0	ITA, LAT	68 728	7.12% ☆
Latin Humanisitic manuscript 17th century	LAT	11 511	4.42% ☆

Figure 2. The 17 Latin Text Recognition Public Models available on Transkribus.

In this first phase of the experiment, we decided to measure the time spent in each step of the creation of these *pure models* – from uploading images to *training*, including layout and *Ground Truth* preparation. This data could allow us to consider if Transkribus and HTR-based software were viable options for the project: unless the overall time spent in all phases up to the production of the AI-based transcriptions was significantly lower than the time needed for human-created transcriptions, this tool could not be considered viable for the project, especially taking into account it is a paid software. We produced 26 models in this first phase, opting for a system of *Ground Truth* creation based on the insertion of *direct transcriptions* – i.e. Latin text with expanded abbreviations and free of graphic signs – which proved to be an efficient system for the specific purposes of the project. The other option, an input system of *diplomatic transcriptions* – i.e. abbreviated Latin text containing graphic signs more coherent for the relationship between image and text – not only required much more time for the production of *Ground Truth* comparatively, but also revealed incompatibility issues between Transkribus and Unicode platforms specialized in characters of medieval manuscripts, such as MUFI (*Medieval Unicode Font Initiative*).

One model that is particularly representative of the group is the MemoBo, Me. 69, Model 4, created with 50 pages – 44 in the *Train Set* and 6 in the *Validation set* – in all 16243 words.

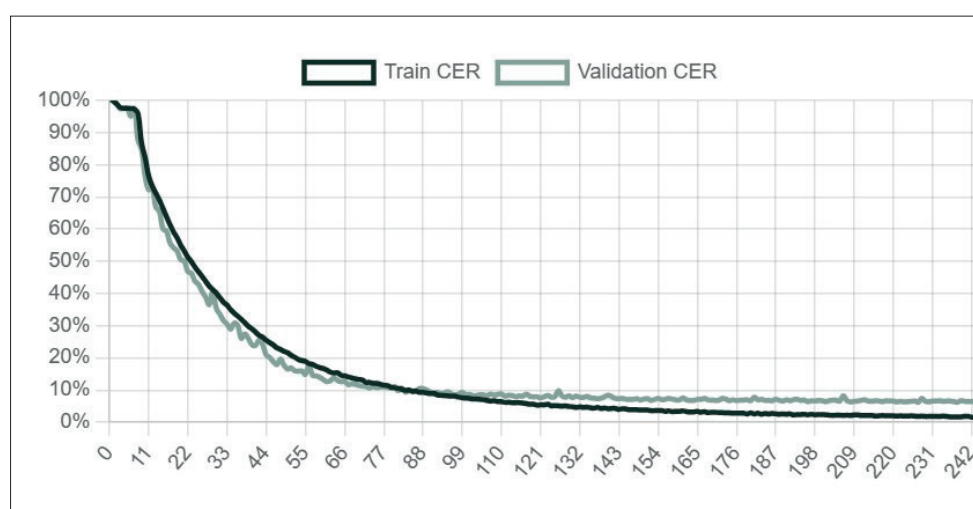


Figure 3. The Learning Curve of MemoBo, Me. 69, Model 4.

We spent roughly 30 hours in the production of this model from scratch and its parameters are the following: 1,60% of CER (Character Error Rate) on train set² and 6,30% of CER on validation,³

² The Character Error Rate (CER) on train set consists of the numbers of errors in terms of characters produced by the software when trying to learn from the material – the *Ground Truth* – provided to the machine. The percentages given represent the portion of the text and of the operations used to created it – for example, expanding abbreviations – that the machine did not understand. Therefore, a CER on train of 1,60% means the machine understood and took into consideration 98,4% percent of all the elements and processes present in the *Ground Truth* given to it.

³ The Character Error Rate (CER) on validation set consists of the number of errors in terms of character produced by the software when trying to apply everything it learned from the *Ground Truth* in producing its own transcription. To establish this rate, the software takes one a pre-selected page of the *Ground Truth*, erases it completely then attempt to transcribe

which allow us to consider this model as already optimal according to the Transkribus guidelines. In practical terms, it means that a transcription using this model with this specific material – a single-hand script – contains one mistake every 15 words.

This was an excellent first outcome, especially considering the CER parameters provided by the guidelines of the software, as becomes clearer in the table below:

	CER	Training Data
<i>Printed text</i>	0,5-2%	~ 5.000 words / 25 pages
<i>Single hand simple writing</i>	2-4%	+10.000 words / 50 pages
<i>Several hands – all seen during training</i>	4-6%	several 10.000 words training data / several 50 pages
<i>Many hands from same period and region – not all seen during training</i>	6-8%	+100.000 words / more than 500 pages

Table 1. Transkribus' guideline parameters for evaluating the CER on Validation

Despite this excellent outcome, these 26 models demonstrated no versatility whatsoever, with attempts to apply them individually to other units of the collection composed by different notaries failing considerably, as expected.

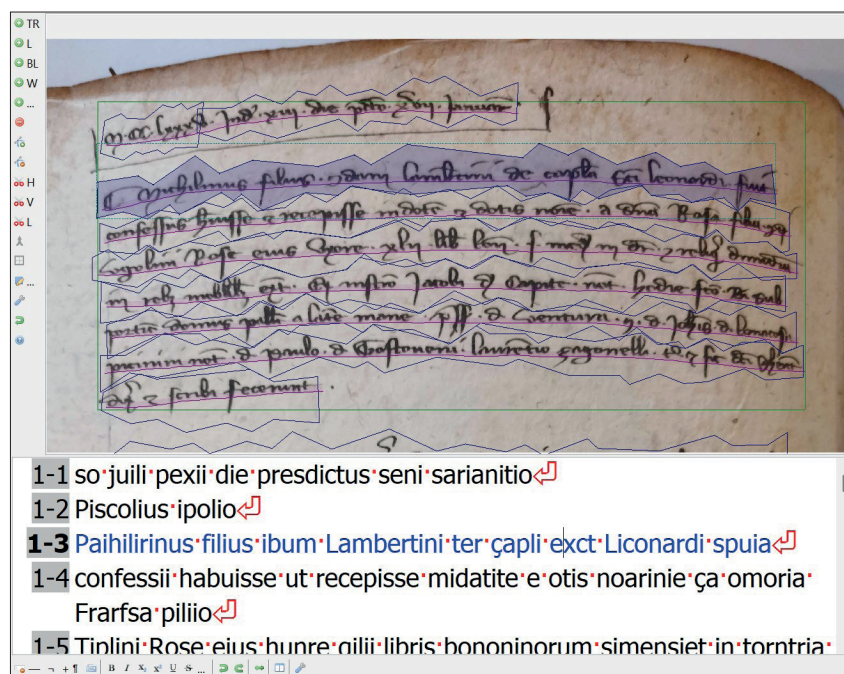


Figure 4. Attempt to apply the MemoBo, Me. 69, Model 4 on a different volume of the *Memoriali*.

it. The resulting transcription is then compared to the original to determine how many mistakes the machine made. This rate is usually considerably higher than the CER on train, since it is an active and generating operation. Therefore, a CER on validation of 6,30% means that the software produced correct transcriptions compared to the selected examples of the *Ground Truth* 93,70 % of the time.

The second stage of the experiment initiated the merging of these 26 models in different combinations to create the first multi-hand *mixed models*. Interestingly, the resulting models started to demonstrate incredible flexibility as early as we introduced a second hand to a previously *pure model*. An example is the MemoBo, Me. 65 and 69, Model. 3, composed of 50 pages of each of these units – both of them from the late thirteenth-century – and presenting a CER on train of 4,90% and a CER on validation of 9,70%.

MemoBo, Me. 65 and 69, Model. 3	
N° of GT pages	100
CER on train	4,90%
CER on validation	9,70%
Hours of work	50

Table 2. Parameters of a MemoBo *mixed model*

These rates are higher than the individual *pure models*, but still optimal, especially considering that this model is composed of mixed material in terms of resolution – half of these 100 images are professional and the other half are amatorial reproductions. An attempt to apply this dual-hand model on material from the first years of the fourteenth century – units not included in the production of its *Ground Truth* – resulted on some perfect lines of transcription even on amatorial reproductions, allowing us to reach an initial conclusion: the software has less difficulties in handling the late medieval abbreviation system than the significant variation of notarial hands inside the collection. This outcome indicated a promising path to the experiment and allowed us to move forward trying on more complex units of the *Memoriali* collection dating from the second half of the fourteenth century.

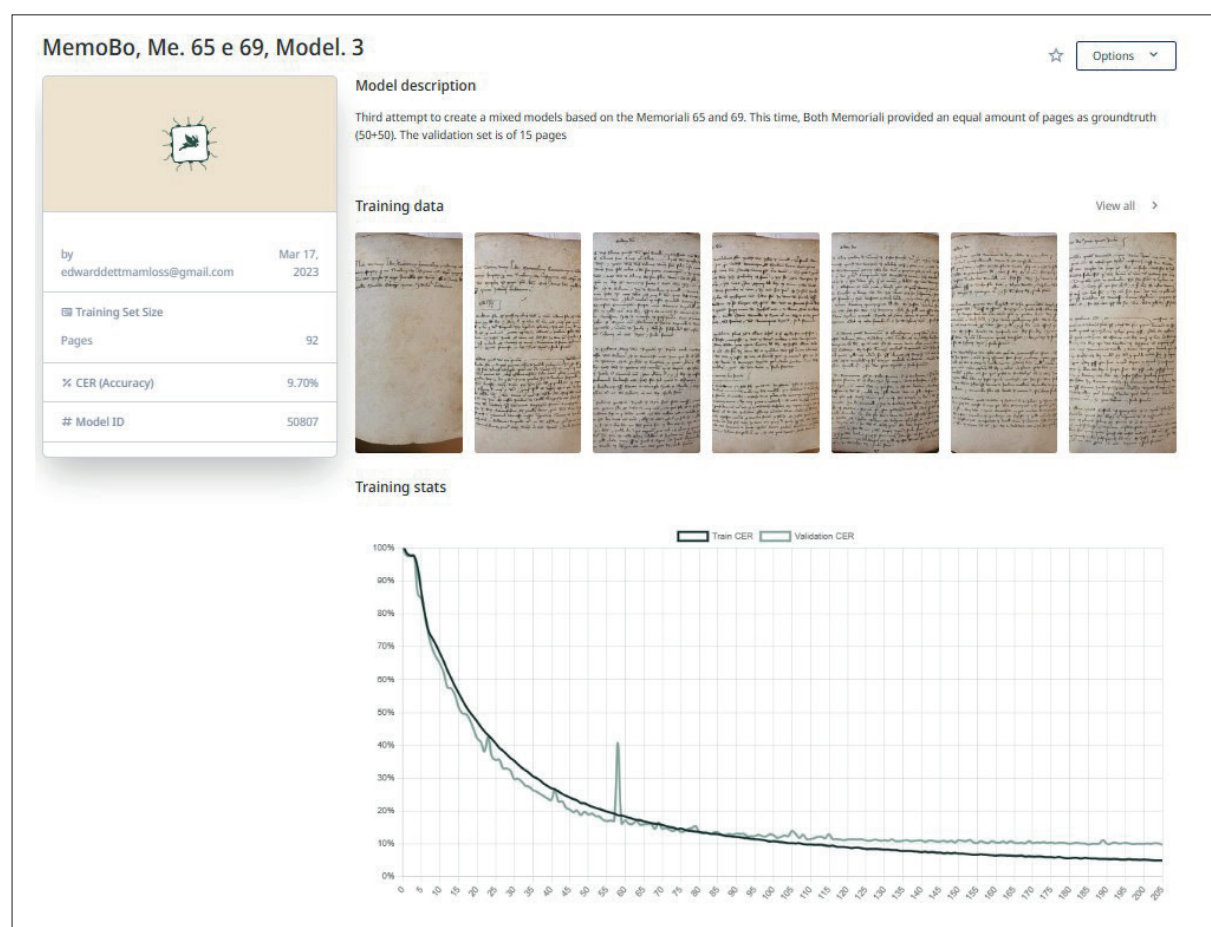


Figure 5. Training set and Learning Curve of MemoBo, Me. 65 and 69, Model. 3.

The third phase of the experiment drew from the previously created *pure models* and these *mixed models* to face the most challenging part of the *Memoriali* collection: the fifteenth-century units, which sometimes contain 40 different hands in a single volume, some of them represented only in fragments of a few *folios* each. The only method capable of treating these units was to invest on *Mega Models*, with large quantities of *Ground Truth*. Apart from substantially augmenting the number of pages and words, the creation of these *mega models* also counted heavily on the previous experiences by the direct inclusion of the *mixed models* above-mentioned on their training. The first one of this category was the MemoBo, Me. 320 and 321, Model 13, created with a *Train Set* of 310 pages (385712 words) and a *Validation Set* of 47 pages (15% of the *Train Set*), proportionally selected from the 40 notarial hands that compose both of the fifteenth-century volumes of the series. We chose to include a combination of *mixed* and *pure models* previously created on thirteenth- and fourteenth-century volumes of the *Memoriali* (MemoBo, Me. 1 and 58, Model. 3 and MemoBo, Me. 298, Model. 5) and the resulting parameters were a CER on *Train* of 4,27% and a CER on *Validation* of 5,78%, with the peculiarity of a *Learning Curve*⁴ that started its learning process already under a scale of 50%.

⁴ A *Learning Curve* represent the learning process of the machine based on the *Ground Truth* provided to it. The X-axis

MemoBo, Me. 320 and 321, Model. 13	
N° of GT pages	310
CER on train	4,27%
CER on validation	5,78%
Base models	MemoBo, Me. 1 and 58, Model. 3 MemoBo, Me. 298, Model 5

Table 3. Parameters of a MemoBo *mega model*

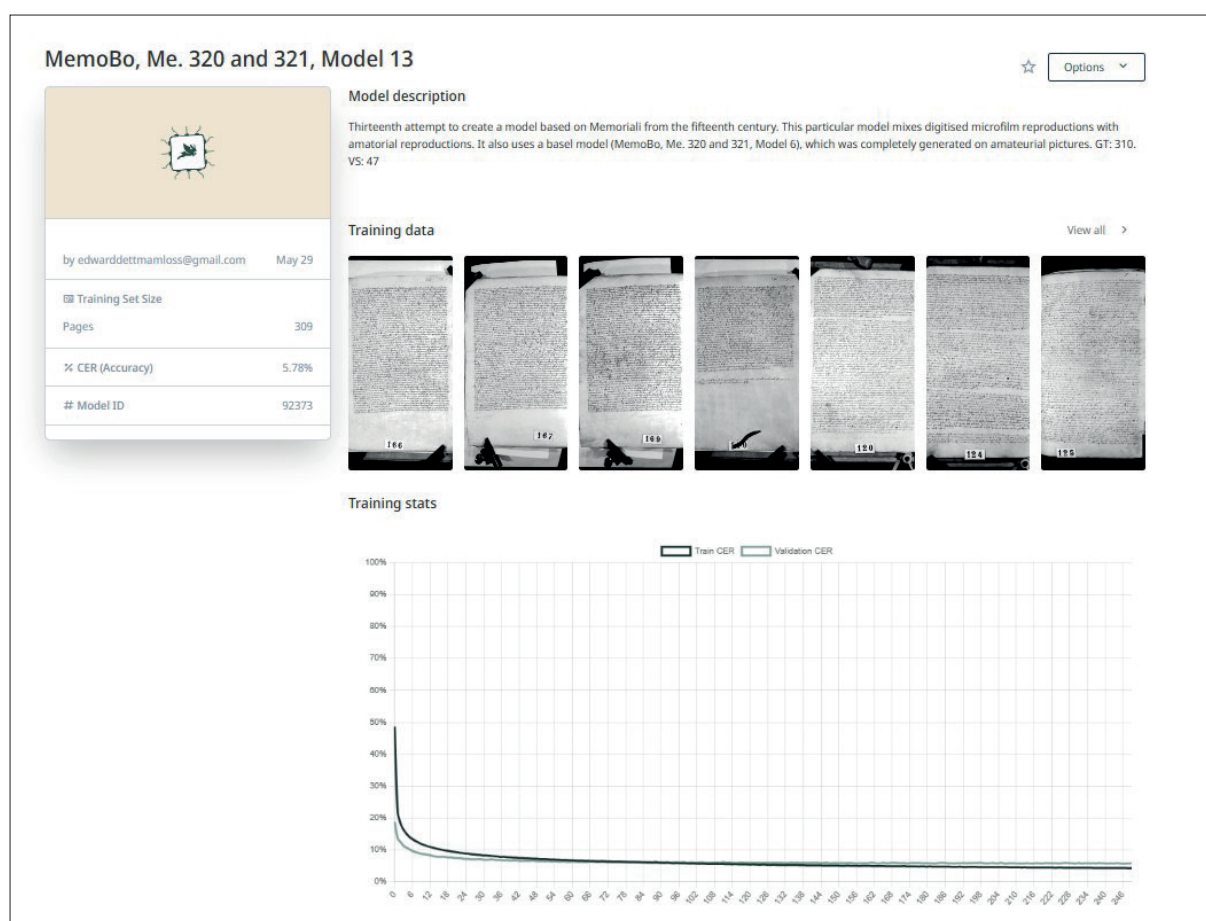


Figure 6. Training set and Learning Curve of *MemoBo, Me. 320 and 321, Model. 13*.

represent the number of times the software read through the *Ground Truth* material, also called cycles. In this example, the software processed the entire material from beginning to end 250 times, doing thus 250 cycles of learning. The Y-axis represents the Character Error Rate in Train and the Character Error Rate in Validation in each of these cycles, allowing the user to follow closely the accuracy of the machine in each of these learning attempts.

In practical terms, the precision of these *mega models* was such as to allow us, for the first time since the beginning of the experiment, to try applying these models produced exclusively with Bolognese documentation on other fifteenth-century notarial records from the Italian peninsula. Bologna and its notaries highly influenced notarial practices in the Late Middle Ages, with notaries, such as Rolandino Passeggeri, who even defined the canons of the notarial practice more generally in the period (Tamba 1998). His *Ars Notarie* was a true best-seller even north of the Alps, and traces of Rolandino's innovative protocols and *formulae* resonate in documentation produced in many cities of modern Southern France and Germany (Tamba 2002). The idea was to verify how much this theoretical influence was translatable in terms of HTR, since *mega models*' main characteristic is their ability to overcome hand variety inside documentary *corpora* of the same chronology. If in terms of format and documental structure other cities followed the Bolognese example, models based in large quantities of *Ground Truth* produced on these documents should, in theory, be able to read successfully records from elsewhere.

We selected Genova and Perugia for this specific part of the third phase of the experiment, since both cities are the ones with the largest collections of surviving notarial records of the peninsula (Macchiavello and Rovere 2018, 43-44; Bartoli Langeli 2006). This ongoing attempt has already demonstrated a lot of potential: for example, attempts to apply the same MemoBo, Me. 320 and 321, Model 13 on the production of notaries, such as the Genoese Antonio de Inghibetis de Castro, without including any additional *Ground Truth* – either from this specific geographical context nor from this specific notary – produced transcriptions with precision verified with Transkribus' tool of *compute accuracy* (still only available on the Expert version) on the scale of 70%, especially applying *compare text versions*. This promising outcome signals larger *mega models* than the MemoBo, Me. 320 and 321, Model 13 could become reference *base models* to explore late medieval notarial documentation on a national or even international level.

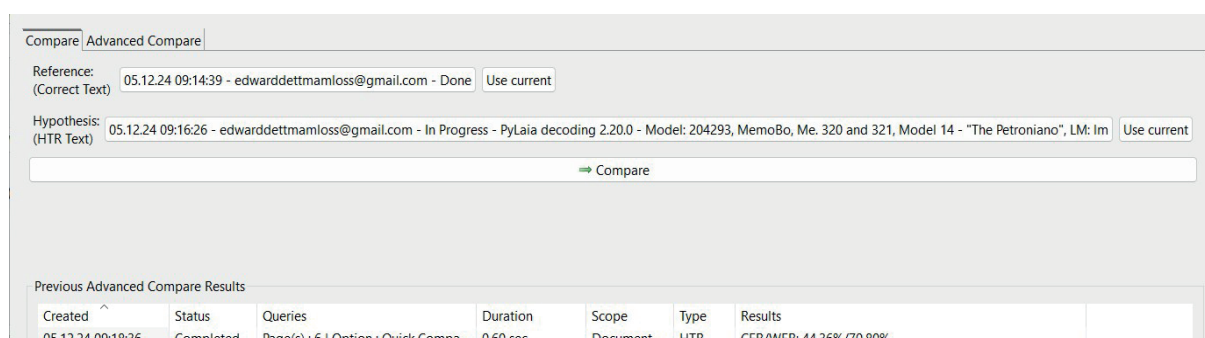


Figure 7. Compare text versions function applied on Antonio de Inghibetis records.

While investing time on the creation of these *mega models*, we also started exploring other functionalities of Transkribus that could have been useful for the project. Since we are interested in extracting specific information from these records, the existence of a customizable tagging tool in Transkribus could offer us the possibility to expedite the input of information inside our database.

5. Transkribus and Tagging

While HTR is the main feature of Transkribus, the software also allows users to mark (*tag*) specific parts of the uploaded images and generated text both during pre-processing and post-processing. The function of this tagging feature is varied and it operates in two levels. First, tags can be *structural*, meaning that they intervene directly on the image section of Transkribus and on the different *Text Regions*⁵ created during layout generation (fig. 8). Second, they can be *textual*, interfering – as its name suggests – directly on the transcriptions, and while *structural tags* cannot be juxtaposed, it is possible to overlay *textual tags* in many different levels (fig. 9).

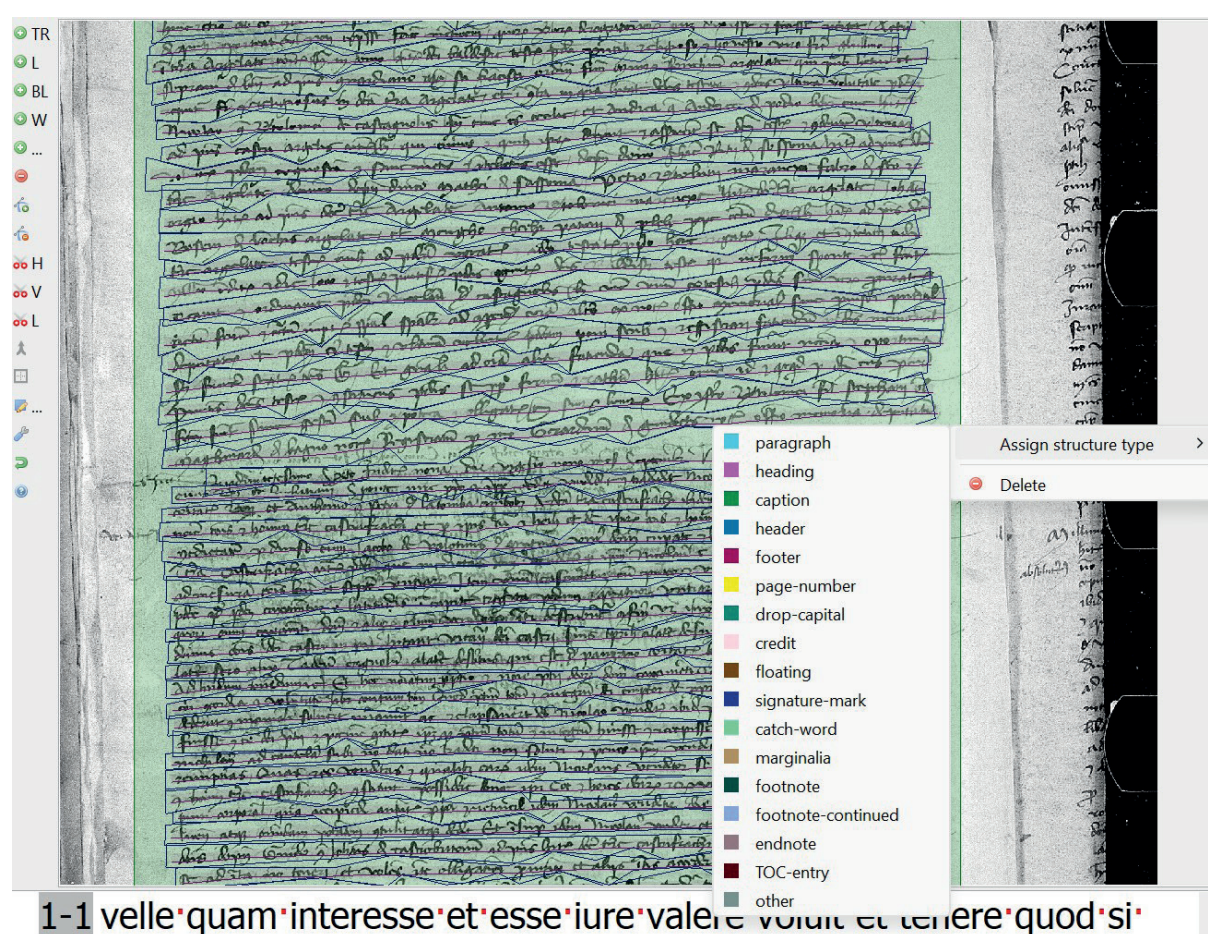


Figure 8. Text Regions and Structural Tags.

⁵ A *Text Region* (TR) is the space within an image that contains the text that user considers to be relevant both for the machine to learn and to transcribe. It does not need to correspond to all text existent inside an image, as it can be seen in Figure 8, where a lateral column of text coming from a different folio was not encased in a text Region on purposes. Elements not included in a Text Region are not even seen by the software.

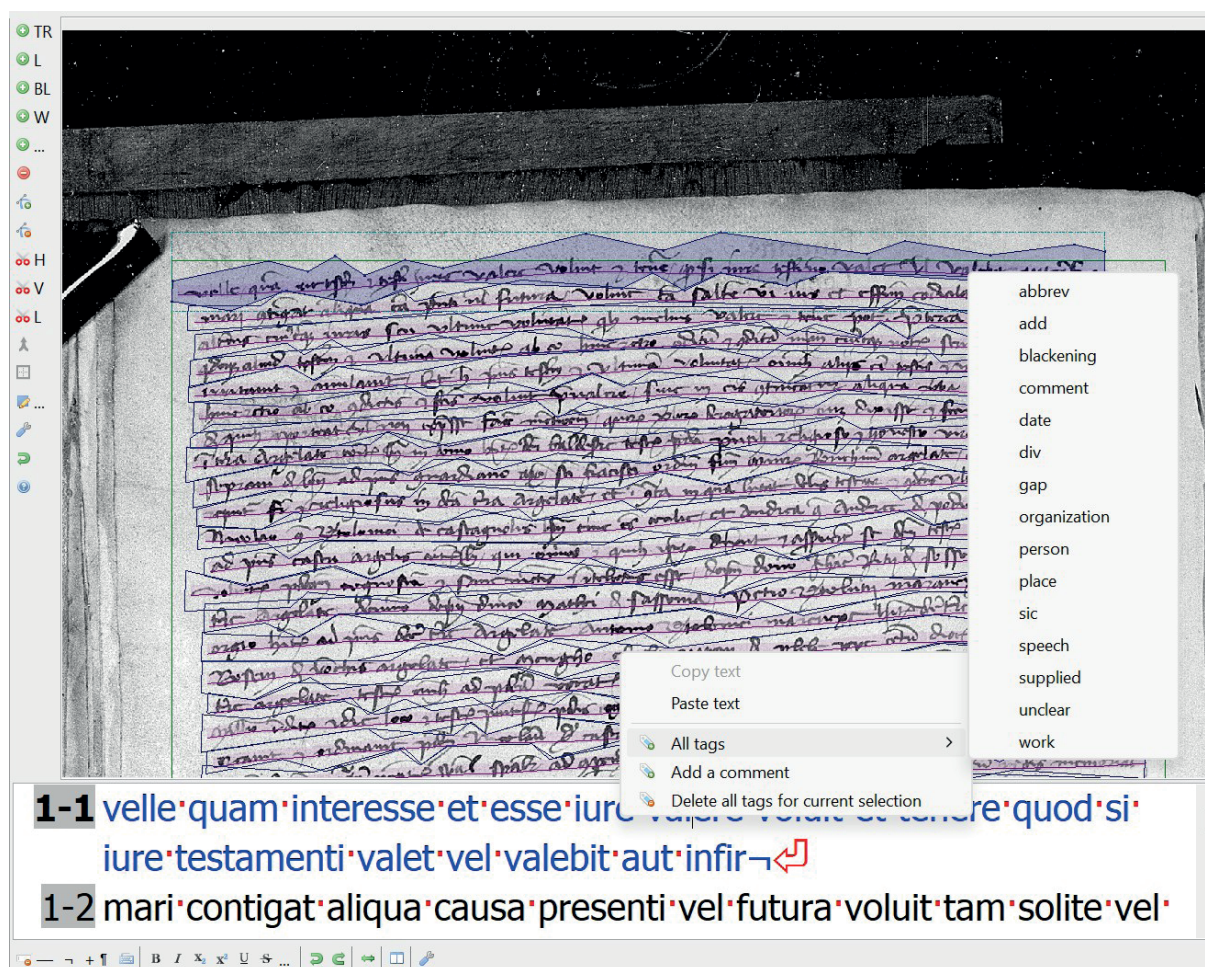


Figure 9. Transcriptions and Textual tags.

Tags leave a substantial imprint on the XML version of exported transcription files, which the user only has access to after properly exporting the material from Transkribus – an important limitation, as highlighted in sections 3 and 4 of the paper. *Structural* tags create and label *Anonymous Blocks* (<ab>) while *textual* tags only operate inside *Line Breaks* (<lb>). While the software offers users with a package of pre-established tags in both categories, it is possible to customize this tool without imposed limitations on the number of tags. While these customised tags must be inserted individually and by hand in the software, once they are created they become an accessible part of the feature in every page, document and collection, turning them into recursive elements (figure 10).

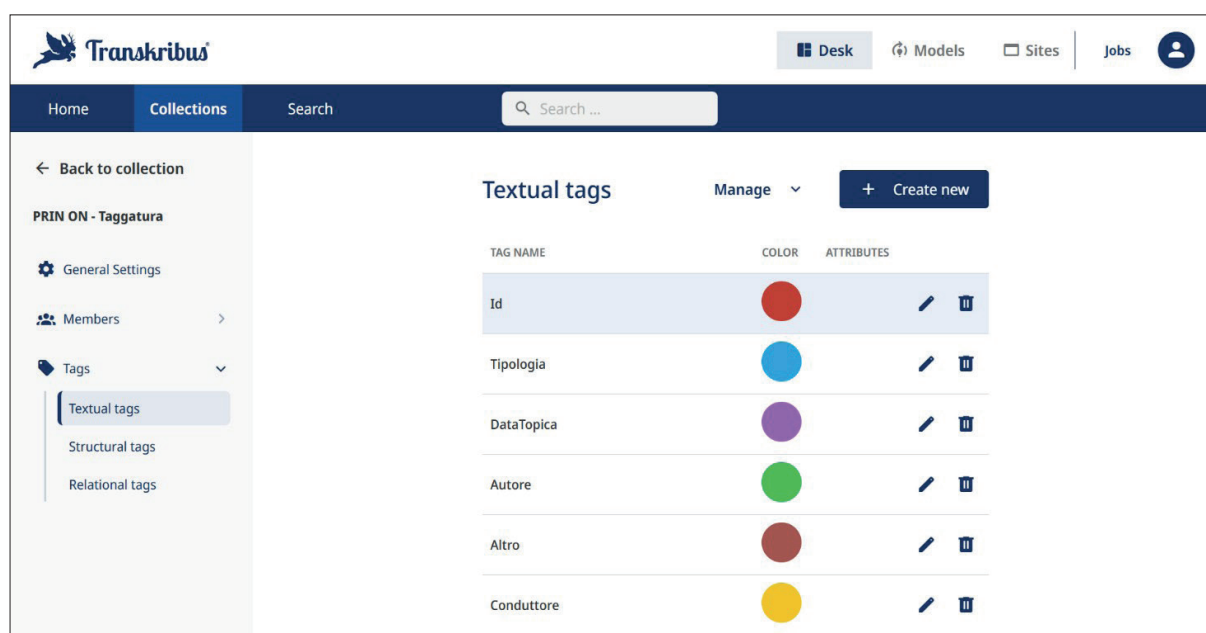


Figure 10. Customised tags as an accessible feature to all collections⁶

Creating a customized tag system

Considering these specificities, the first step to create a customised tagging system on Transkribus that somehow mirrored the categories of our xDams database was to render the software's *Text Regions* circumscribed to each individual act contained in one *folio* of the *Memoriali*. The easiest way to do so was to slice a single macro-*Text Region* automatically generated with a standard Transkribus layout model, posteriorly adding to each of these resulting different *Text Regions* a single *structural tag* purposely created and called <Deed> (record). As mentioned above, the record constitutes the basic unit of our xDams database, therefore, the *structural tag* <Deed> generated the base within which we built our tagging system. Records, however, are not circumscribed to a single *folio*, sometimes continuing on the *verso* side of the manuscript, or – especially in the case of later volumes of the *Memoriali*, produced after 1321 – through several folios. Transkribus structural tags are circumscribed to a single photo, not allowing users to create tags that run from one picture to the other. We came across this limitation with the first attempt of exporting the material into the xDams database – the more technical aspects of this operation in its different phases are better described in the next section – and solutions to overcome this restriction, thus respecting the documentation's basic unit, surfaced only after continuous exchange of ideas and collaboration among the members of the project who are responsible for the Transkribus direct experimentation, in one side, and those more involved with the database and its population in the other. We did not operate in separate compartments; therefore, the creation of each new tagging element was

⁶ The screenshots are taken from the applications used for the project, which are currently set to Italian. To facilitate readability, we have translated the terminology into English; however, it was not possible to translate the applications themselves at this time. In the screenshot in Figure 7, “tipologia” stands for “type”, “data topica” stands for “topical date”, “autore” stands for “author”, “altro” stands for “other” and “conduttore” stands for “conductor”.

followed by an importation attempt, which highlighted interruptions in the basic structure of the act. A strategy to solve them was to create another structural tag <DeedIntegration>. In order to reconduct a certain structural tag <DeedIntegration> to its correspondent <Deed>, we started intervening and populating these resulting *Text Regions* with *textual* tags. The first one we created was a *textual* tag <id>, which, once placed on a user introduced number in both the structural tag <Deed> and <DeedIntegration>, could easily produce an unequivocal element – necessary criteria, again better explained in the next section, for proper importation.

After the *textual* tag <id>, we created other tags on the textual level that mirrored our database and its inherent “record” categories. For example, we invented the tag <TopicalDate> to mark the portion of the automatically transcribed text that corresponded to this field of our database and so on.

During our first attempt to implement all of these textual tags, we noticed that whenever the information changed from one line to another, Transkribus ruptured our *textual* tags into two different ones of the same category. This peculiarity forced us to operate with the same strategy as the one chosen for the creation of the structural tag <Deed>, which required the creation of a <DeedIntegration> tag. Therefore, all of the *textual* tags developed during this first attempt also included their “integration” version. This case is also better explored in greater detail in the next section.

Another problem that emerged was connected to all information regarding the people involved in the transactions described in the *Memoriali*. The multiplicity of individuals involved in a single deed, together with the intent of the *MemoBo*’s project to classify very specific details from each one of them, imposed a series of potential unwanted juxtapositions of general *textual* tags, such as <name> and <surname>, which could be applied to each and any of the people contained in the acts. A solution to this problem concerning the absence of univocal entries was to classify individuals inside a deed according to their function, and then overlay all these general textual tags to this specific function. We called this operation of overlaying two levels of *textual* tags as the “*Tag on Tag system*”.

The Tag on Tag system

When tagging the various deeds inside Transkribus following this system, we first inserted a tag corresponding to one of the 18 functions with which we classified agents in our xDams database (fig. 11). Once applied, we proceeded nesting inside this first macro-textual tag concerning the *function* all of the more general ones, such as <name> and <surname>, carefully paying attention to not juxtapose their extremities in order to not corrupt this hierarchical scheme (fig. 12).

MODIFICA

IDENTIFICAZIONE E DESCRIZIONE

INDICI NOMI

INDICI LUOGHI

NOTE E COMPILAZIONE

SUPERIORE

SALVA

SALVA E CHIUDI

CHIUDI

genere

nome

cognome

età

provenienza o residenza

luogo 1

code luogo 1

luogo 2

code luogo 2

qualifica

ente di appartenenza/riferimento

condizione

mestiere

funzione

...

arbitro

autore

destinatario

conduttore

confinante

creditore

curatore

debitore

erede

esecutore

fideiussore

legatario

presente

procuratore

sostituto testamentario

testimone

titolare precedente

tutore

altro

...

APRI

note

INSERISCI

RIMUOVI

Figure 11. The list of 18 functions of agents in the xDams platform.

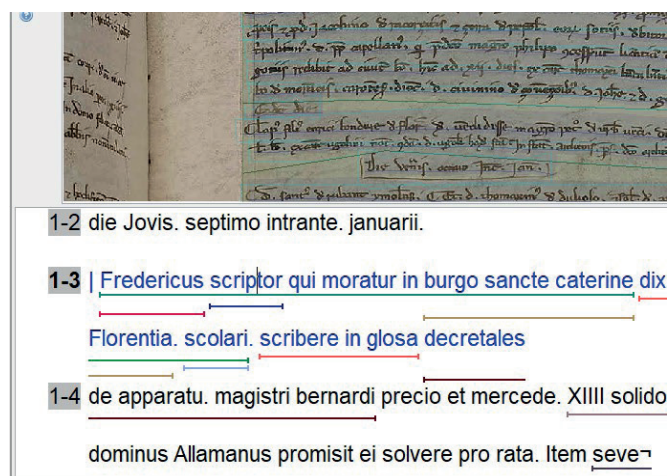


Figure 12. "Tag on Tag" system: nesting tags under function tag.

Since many different agents can exercise the same function inside a certain record, we also introduced the previously mentioned tag <id> inside each of these nests organised according to function in order to preserve their univocity (fig. 13 and 14).

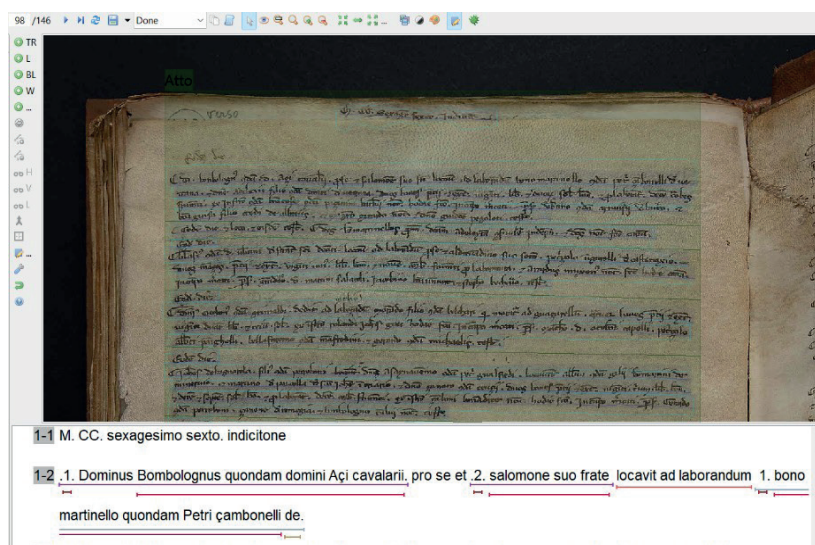


Figure 13. “Tag on Tag” system: textual tag <id>.

```

<Witness>
  <id>1</id>
  . dominus
  <Name>branchaleonus</Name>
  <Surname>de lambertacius</Surname>

</Witness>
et dominus
<Witness>
  <id>2</id>
  <Name>madius</Name>
  <Surname>de lambertacius</Surname>

</Witness>
<lb facs="#facs_105_tr_1_tl_7" n="N006"/>
et dominus
<Witness>
  <id>3</id>
  <Name>guillelmus bocatiis</Name>

</Witness>
et dominus
<Witness>
  <id>4</id>
  <Name>Lambertinus quondam domini mulinaroli</Name>

</Witness>
et
<Witness>
  <id>5</id>
  <Name>marchisinus</Name>
  <Surname>de datis</Surname>

</Witness>
<Witness>
  <id>6</id>
  <Name>guidonus</Name>
  <Craft>sartor</Craft>

</Witness>
et
<Witness>
  <id>7</id>
  <Name>Guilleminus</Name>

</Witness>
<lb facs="#facs_105_tr_1_tl_8" n="N007"/>
<WitnessIntegration>

```

Figure 14. Tag <id> on Transkribus-generated XML file.

Creating and tagging a “level object”

Our xDams database contemplates all properties, assets and rights contained inside a late medieval notarial record as a separate hierarchical level, as described in the previous session. This peculiar configuration imposed a different logical scheme for tagging all portions of the *Memoriali* records that concerned this specific information, since it rendered necessary to detach elements that appear in the original records as united. In order to separate these assets into a portion of the record that could be easily read in the posterior phase of importation – described in detail in the next topic – we added yet another level of tagging in our “Tag on Tag system”: a macro textual tag <ObjectLevel>, which should nest the three textual tags correspondent to each of the asset categories of our database – <ImmovableProperty>, <MovableProperty>, <IntangibleObject>. Each of these asset categories then hosts a general *textual* tag called <Description> – a solution found to accommodate all the differences and varieties of elements that compose the set of information regarding this specific level. The juxtaposition of the three tags hierarchically renders the following structure (fig. 15).

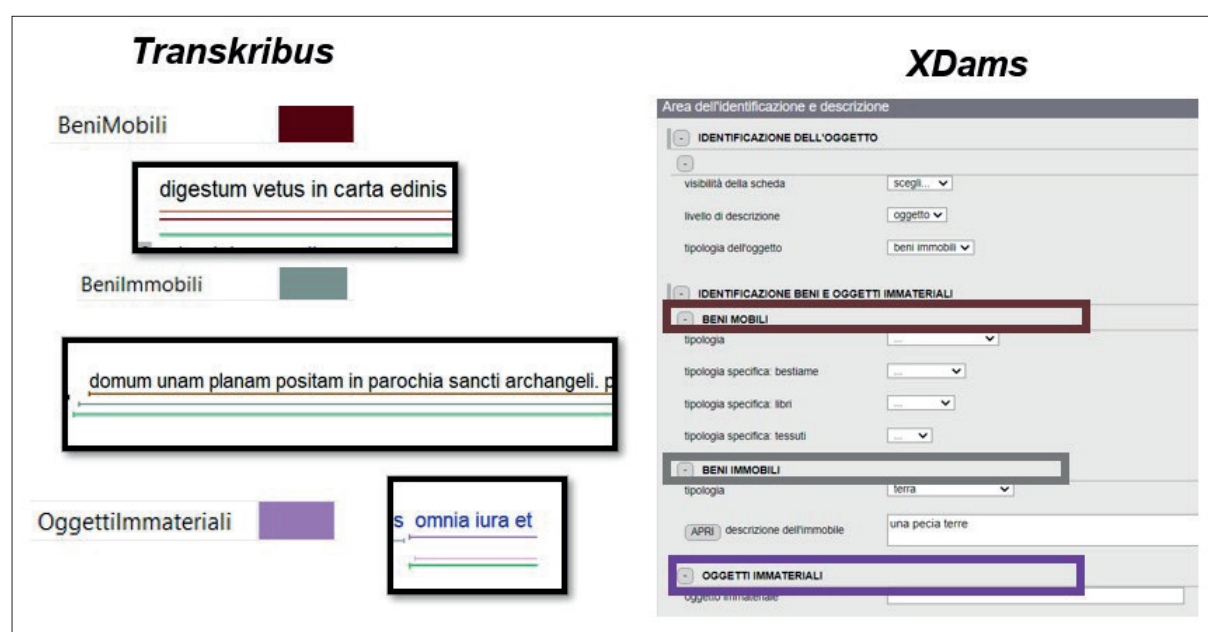


Figure 15. Constructing the level object – comparison between Transkribus and xDams categories.

Once this structure was completed, we then moved on to the first attempts of direct importation from Transkribus to xDams.

6. Importing Transkribus XML into xDams

To obtain a file compatible with the *Memorialis*' layout, it was necessary to carry out analysis and mapping work, which required iterative interventions to achieve an optimal result.

The process was divided into the following phases:

- analysis of Transkribus XML and mapping
- import procedures
- data validation.

The analysis of Transkribus XML and mapping

The analysis was a preparatory and essential phase for the successful completion of the subsequent steps. The XML file generated by Transkribus, which employs a schema provided by the software, has the following structure:

- `/teiHeader/fileDesc/titleStmt/title/text()`: xPath where the registry information is stored;
- `/text/body/div/pb`: each `<pb>` element represents a page contained within a registry;
- `/text/body/div/ab[xx]/@type`: each `<ab>` element represents an act, contained within a page;
- `/text/body/div/ab[xx]/lb[xx]`: each `<lb>` element represents a row, contained within an act.

Given the configurability of Transkribus, tagging was carried out, as far as possible, based on the scheme already used by the *Memoriali* project, as described in the previous section.

During the analysis phase, several critical issues emerged:

1. Transkribus creates a `</TEI/text/body/div/pb>` tag for each physical page it encounters, and it closes this tag before moving on to the next page. This tag also contains the `"@xml:id"` attribute, which specifies the name of the corresponding image. Simultaneously, one or more `</TEI/text/body/div/ab[xx]>` tags are created, identifying a "deed", which are still closed when moving to a new physical page. Thus, two situations may arise: either pages that contain multiple documents, represented by different `</TEI/text/body/div/ab[xx]>` tags opened and closed before proceeding to the next `</TEI/text/body/div/pb>` tag, or a "deed" that spans two or more pages, exceeding the limit of the physical page. The issue lies in the fragmentation of a single document into multiple parts, thereby losing its unity.
2. Transkribus renders the two nodes `</TEI/text/body/div/pb>` and `</TEI/text/body/div/ab>` as siblings and parallel to each other, without any hierarchical relationship: this causes issues when nesting the physical page information during procedural processing.
3. Each deed contains multiple `<lb>` lines, and each line can include several elements that map the information contained in the deeds, such as people and objects involved in the transactions. It is possible that the description of an element does not end at the physical end of the line, so the data is stored on the next line, creating a new element that would overwrite the previous one during procedural processing if not handled correctly. The issue here is the fragmentation of the information across multiple tags of the same type, leading to transcription errors.

As mentioned in the previous section, it was, therefore, necessary to adjust tagging in Transkribus by introducing an <integration> tag, which allows documents spanning different pages and elements whose descriptions exceed the physical line to be linked. This linkage is possible by the use of an <id>[n]</id> tag, placed both in the main element and in the supplementary one (fig. 16 and 17).

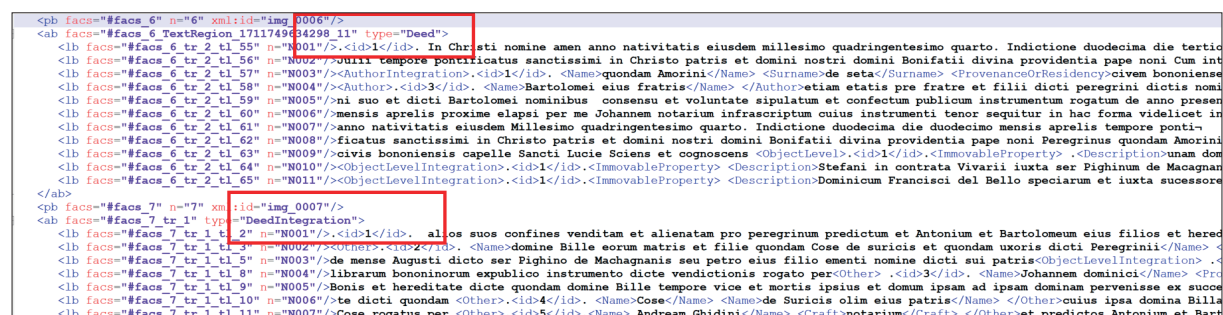


Figure 16. Linkage of Deeds spanning different pages in XML.

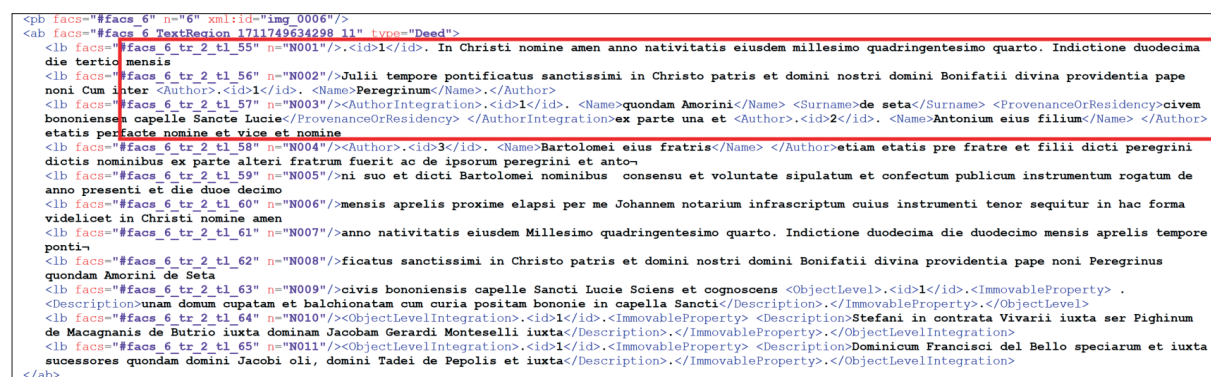


Figure 17. The linkage with <id> element.

This aspect deserves greater attention, as it is closely related to the transformation process, as seen below.

Technologies

For the analysis, processing, and importing of the data produced by Transkribus, we chose to use the Java programming language in conjunction with the XMLBuilder library, a parser developed by Regesta.exe. This library efficiently handles CRUD (create, read, update, delete) operations on XML-formatted resources as well as the migration of data into the designated database. As the final repository, we used xDams, an Open Source document management platform developed in ASP (Application Service Provider) mode, which uses the Web for data access and interaction. It is designed for the management and integrated use of multimedia historical archives.

Phases of the procedure

The first step necessary for the procedure was to process the XML version generated by Transkribus, as described previously, in order to create a new version with some modifications, particularly related to pagination and the structure of “deeds”.

The hierarchical “sibling” relationship between the <pb> and <ab> elements is not an optimal structure for processing, which required intervention by first removing the </TEI/text/body/div/pb> nodes from their original position and reinserting them, hierarchically nested, within the corresponding </TEI/text/body/div/ab[xx]> tags as their “children”. This way, by iterating only through the </TEI/text/body/div/ab[xx]> nodes, the information related to the reference pages is directly accessed.

Next, we addressed the management of the multi-line and multi-page issues: in both cases, Transkribus generated multiple identical nodes where it stores information related to lines affected by line breaks or page changes. This behaviour is also not ideal for procedural processing. We resolved it by introducing two interrelated concepts: the first is “integration”, a system we use to identify nodes referring to the same resource (block of lines, block of pages, etc.); the second is the <id> tag, which is used as a key to recognize and link those nodes. Here is a practical example to better clarify the concept:

```
<lb facs="#facs _ 7 _ tr _ 1 _ tl _ 21" n="N014"/>
<ObjectLevel>
  <id>2</id>
  <ImmovableProperty>
    <Description>una petie terre aratorie</Description>
  </ImmovableProperty>
</ObjectLevel>
<lb facs="#facs _ 7 _ tr _ 1 _ tl _ 22" n="N015"/>
<ObjectLevelIntegration>
  <id>2</id>
  <ImmovableProperty>
    <Description>vidate et arborate sex tornaturarum et cum
eo pluri vel minori quod     esse reperiatur posita in guardia terre
funi comitatus bononie in loco gualfundo</Description>
  </ImmovableProperty>
</ObjectLevelIntegration>
```

In this example, we can see how the data contained in the <ObjectLevel/ImmovableProperty/Description/text()> node begins on a specific line, ‘N014,’ but continues on the following one, marked by the <lb> tag with n=‘N015’. To retrieve the values contained in both “Description” nodes, we created the <ObjectLevelIntegration> tag, which acts as an integration of <ObjectLevel>, and linked them through <id> tags, both set to “2” in this case. By doing so, during procedural processing, it becomes easier to loop through the various nodes, retrieve the text in the <Description> nodes, and concatenate them into the string “una petie terre aratorie

vidate et arborate sex tornaturarum et (...)”, which will then be written into the designated *XPath* for that value in the target database schema. The solution just described, regarding the use of ‘integration’ and `<id>`, is applied and repeated across all levels (`<Deed>`, `<MovableProperty>`, `<ImmovableProperty>`, and `<IntangibleObject>`), and involves all the XML nodes, including repeatable ones (`<Author>`, `<Recipient>`, `<Conductor>`, `<Neighboring>`, `<Creditor>`, `<Curator>`, `<Debtor>`, `<Heir>`, `<Executor>`, `<Guarantor>`, `<Legatee>`, `<Present>`, `<Procurator>`, `<TestamentarySubstitute>`, `<Witness>`, `<PreviousOwner>`, `<Tutor>`, `<Other>`, `<ObjectLevel>`) and those that occur only once per “Deed” (`<Type>`, `<TopicalDate>`, `<RecordingNotary>`, and `<ProvenanceOrResidency>`). In this way, we ensure that all portions of resources “split” by Transkribus can be uniquely linked and reconstructed.

This solution, however, introduces another critical issue: Transkribus, and consequently the integrations we added, generate *textual* tags based on the physical order of the manuscript lines, including line breaks. This means that nodes may contain entire words or sentences, but also fragmented sentences or, worse, truncated words, as a result of line breaks. Managing and correctly resolving all possible concatenations automatically is impossible, and human intervention is required to correct eventual introduced errors.

We considered two possible approaches: the first involved processing each retrieved value with a method (`.trim()`) that removes any leading and trailing spaces from each string, then adding a space before each concatenation. The second approach involved inserting a distinctive symbol (in this case, a double pipe, ‘||’), which is easily recognizable and certainly foreign to the rest of the text, before each concatenated string. Although the first approach had a success rate of approximately 70-80%, it made it very difficult to address the remaining 20-30%. For example, consider the word “habitante”, which is interrupted by a line break in the manuscript and concatenated as “habitant”. The second approach, while it “creates errors” in 100% of cases by introducing symbols not present in the original text (e.g., producing ‘hal||bitante’ in the previous example), makes it much faster to locate and manually correct these issues. This may involve simply deleting the chosen symbol to join the two parts of the string, as in the case of “habitante”, or replacing it with a space when necessary. Since post-procedural human intervention was required in any case, we mutually agreed to adopt the second approach, aiming for lower complexity and increased speed for those responsible for this final data cleanup in the database.

Once these preparatory steps are completed, the remainder of the procedure—namely, the actual extraction of data and its writing into the database—proceeds smoothly, consistently following the same methodology. The process works in levels, starting from the highest and moving downward. Initially, all records corresponding to the `</TEI/teiHeader>` nodes, labelled according to the value found in `</TEI/teiHeader[xx]/fileDesc/titleStmt/title/text()>`, are recognized as “registry” type records and saved in the xDams database under the root element of the hierarchical tree.

The next step is processing “deed” type records, identified by the `</TEI/text/body/div/ab[xx]>` nodes. For each of these, the values contained in the various repeatable and non-repeatable nodes are extracted, reconstructing strings when necessary (in the presence of integrations), and then migrated into their respective *XPath* in the historical schema. Finally, the “deed” records obtained are saved in the database, each hierarchically linked to its corresponding “registry” record.

For all “deed” records, the value contained in the `</TEI/text/body/div/ab[xx]/@facs>` attribute is also retrieved and saved, as it will later serve as a key to identify them unequivocally. After processing deeds, the procedure examines the integrations for these same deeds, repeating the steps described above and adding the corresponding information to the already saved deed records, which are identified using a query based on an exact match between the `<id>` tag values. The final step is the processing of “goods/object” type records, which are of a lower hierarchical level and identified by the tag `</TEI/text/body/div/ab[xx]/ObjectLevel>`. Here, too, the data is extracted in the same way, with string concatenation in cases of integration. These records are further differentiated based on whether they are `<MovableProperty>`, `<ImmovableProperty>`, or `<IntangibleObject>`, and saved accordingly as “children” of their respective document, identified using a query on the database via the previously saved ‘@facs’ attribute. At the end of the procedure, we will have a database in xDams with the following structure (fig. 18):

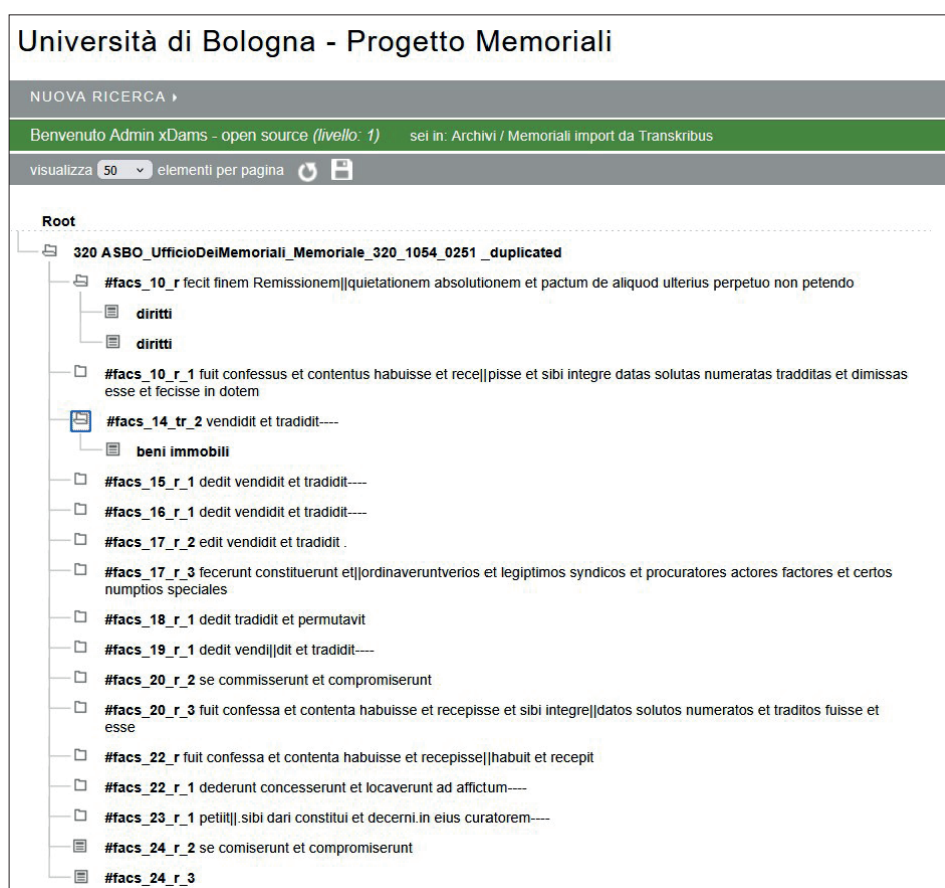


Figure 18. The hierarchical structure in xDams database.

The final step: data validation

At this stage, the results of the procedure are analysed directly within the xDams environment: an initial random check is performed on critical cases (integration cases) to first verify the successful grouping of the ‘integration’ fields.

If issues are identified during the check, the problem must be located, and the mapping and/or transformation procedure corrected.

If the check yields positive results, the data can be handed over to cataloguers, who will complete the descriptions (fig. 19).

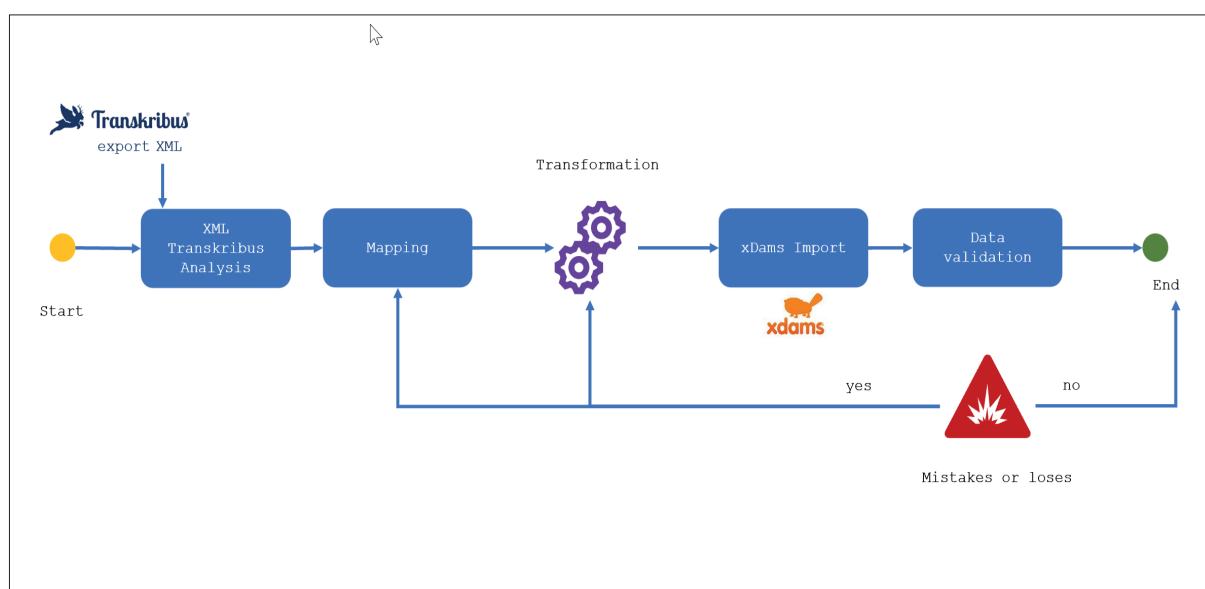


Figure 19. The complete process.

The described process is based on an iterative development cycle: the approach adopted is an iterative and incremental one, with short work cycles and frequent adaptations. The final result has allowed us to obtain a stable schema for reuse and we created a semi-automated loading pipeline.

7. The complete system and final remarks

After this meticulous experimentation, composed, as seen above, by many failed attempts, discussions and retrials, we established a general tagging system capable of producing XML files semiautomatically absorbed by our xDams database with the following structure: 2 *Structural* tags (<Deed> and <DeedIntegration>), 26 *textual* tags relative to the level “Act” (<Author>, <Other>, <Conductor>, < Surname>, <Neighboring>, <Creditor>, <Curator>, <TopicalDate>, <Debtor>, <Recipient>, <Affiliation>, <Heir>, <Executor>, <Guarantor>, <Legatee>, <Occupation>, <RecordingNotary>, <Name>, <Present>, <ProvenanceOrResidency>, <Procurator>, <TestamentarySubstitute>, <Witness>, <Type>, <PreviousOwner>, <Tutor> and <Qualification>) and their 26 integration (<Integration>) versions, plus 5 *textual* tags concerning the level “Object” (<ObjectLevel>, <MovableProperty>, <ImmovableProperty>, <IntangibleObject>, <Description>) and their 5 integration (<Integration>) versions, including, at last, a *textual* tag <id>, useful, as seen before, both to the level “Act” and level “Object”.

The following image gives a visual idea of the application of all these 65 tags in their appropriate cases on Transkribus (fig. 20):

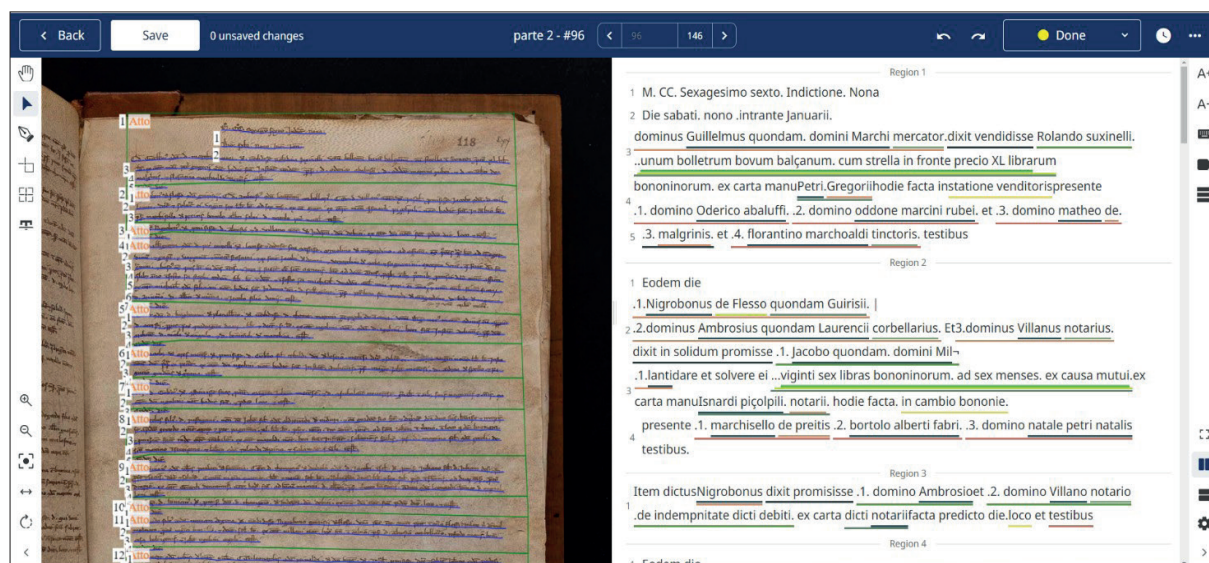


Figure 20. The complete tag system in Transkribus.

As mentioned in the introduction, the goal of this tagging system was not only to test the technological limitations and possibilities of both Transkribus and our xDams database, but also to find solutions that could expedite the general aim of the project to populate the database with complex information extracted from the *Memoriali*. With this objective in mind, all operations described in this paper were meticulously timed to see if and how much faster this system could be in comparison to a human cataloguer that manually inserted all this information in the xDams database. The outcome of this operation is that a regular cataloguer takes approximately 10 minutes to insert manually a complete deed in the xDams database, considering as a parameter the first unit of the *Memoriali* series – the *Memoriali* of Nascimpace Petrizani – marked by the presence of very simple descriptions of all elements relative to the “object” level. Tagging a deed from the same *Memoriali* unit inside Transkribus with all the categories corresponded to the database takes, instead, approximately 3 minutes, but we should also add to this timeframe the time needed to import the resulting XML Transkribus file into the xDams database. To generate a Transkribus XML file that can be acquired to initiate the transformation procedures, the server usually takes up to 1 minute, after which the Regesta.exe can produce a MemoBo XML file that is xDams-compliant and imported into the system. We thus conduct a testing phase, and if the result is satisfactory, we move the output to the working database. The procedure takes approximately 30–40 seconds to process 30 records – equivalent to 1 second per record – including the preliminary stages of correction and hierarchy adjustment. Adding these numbers up, including the time it takes to share a tagged Transkribus XML file among the different members of the project, we could estimate that through this semi-automatic importation system, it would take us approximately 5 minutes to include a deed inside our xDams database. Contrary to the manually inserted records, this deed would require further corrections to bring it to the standards of the database, such as altering names of the deed’s agents to Latin nominative; an operation that we estimate could take up to 2 minutes. The outcome is that in approximately 7 minutes the deed initiated by tagging portions of

a Transkribus HTR generated transcriptions of *Memoriali* has the same standards as the one included by manual cataloguers in as much as 10 minutes. The gain is of nearly 30% of time, which, considering the dimensions of the collection, allow us to consider this strategy, and, consequently, the fruits of our Transkribus/xDams tagging experiment as a viable future for the development of the project.

Apart from the consistent gain in terms of time, it is important to highlight that the adoption of this tagging procedure as a cataloguing method also includes an AI-generated transcription of the material as a final product, with a high level of intervention by human hands applying the tags to the documentation. Though it is not the scope of the MemoBo project to offer full transcriptions of these sources, the resulting texts may be useful for teaching and research, opening new possibilities and applications. There is much that could be said about the cultural implications of rendering available large quantities of information concerning late medieval notarial records; a source, which contrary to normative texts, grant us access to a series of agents and their margins of actions, for example women, not formally included in the channels of political and institutional participation (Piccini, 1996). By applying artificial intelligence tools, thinking on their limitations and possibilities for this specific sort of serial documentation, our project aims to contribute to providing materials to scholars interested on tackling such topics, which could transform our understanding of the dynamics in large communes of the late Middle Ages, such as late medieval Bologna.

References

- Bartoli Langeli, Attilio. 2006. *Notai: scrivere documenti nell'Italia medievale*. Roma: Viella.
- Bruno, Giovanni, Francesca Delneri, Tommaso Duranti, and Fabiana Guernaccini. 2023. "Il Progetto MemoBo: sinergie e nuove sfide a partire dai Memoriali bolognesi." *DigItalia* 18: 150-63.
- Continelli, Luisa. 1988. *L'archivio dell'Ufficio dei memoriali. Inventario. Volume I Memoriali, 1264-1463. Tomo I, 1265-1333*. Bologna: presso l'Istituto per la storia dell'Università.
- Hodel, Tobias. 2022. "Supervised and Unsupervised: Approaches to Machine Learning for Textual Entities." In *Archives, Access and Artificial Intelligence: Working with Born-digital and Digitized Archival Collections*, edited by Lise Jaillant, 157-78. Bielefeld: Bielefeld University Press.
- Macchiavello, Sandra, and Antonella, Rovere. 2018. "The Written Sources." In *A Companion to Medieval Genoa*, edited by Carrie E. Benes, 27-48. Leiden-Boston: Brill.
- Piccinni, Gabriela. 1996. "Le donne nella vita economica, sociale e politica dell'Italia medievale." In *Il lavoro delle donne*, a cura di Angela Groppi, 5-46. Roma-Bari: Laterza.
- Tamba, Giorgio. 1998. *Una corporazione per il potere: il notariato a Bologna in età comunale*. Bologna: Clueb.
- Tamba, Giorgio, ed. 2002. *Rolandino e l'ars notaria da Bologna all'Europa: atti del Convegno Internazionale di Studi Storici sulla Figura e l'Opera di Rolandino. 9 - 10 ottobre 2000*. Milano: Giuffrè editore.