International Journal of Digital Humanities (2019) 1:235–250
https://doi.org/10.1007/s42803-019-00013-9

RESEARCH ARTICLE

Tracking the evolution of translated documents: revisions, languages and contaminations

Gioele Barabucci [1]

Published online: 28 June 2019

Abstract

Dealing with documents that have changed through time requires keeping track of additional metadata, for example the order of the revisions. This small issue explodes in complexity when these documents are translated. Even more complicate is keeping track of the parallel evolution of a document and its translations. The fact that this extra metadata has to be encoded in formal terms in order to be processed by computers has forced us to reflect on issues that are usually overlooked or, at least, not actively discussed and documented: How do I record which document is a translation of which? How do I record that this document is a translation of that specific revision of another document? And what if a certain translation has been created using one or more intermediate translations with no access to the original document? In this paper we addresses all these issues, starting from first principles and incrementally building towards a comprehensive solution. This solution is then distilled in terms of formal concepts (e.g., *translation, abstraction levels, comparability, division in parts, addressability*) and abstract data structures (e.g., *derivation graphs, revisions-alignment tables, source-document tables, source-part tables*). The proposed data structures can be seen as a generalization of the classical evolutionary trees (e.g., *stemma codicum*), extended to take into account the concepts of translation and contamination (i.e., multiple sources). The presented abstract data structures can easily be implemented in any programming language and customized to fit the specific needs of a research project.

Keywords Revision control for translated documents · Independent evolution of translated documents · Data structures for stemma codicum · Multi-language stemma codicum

✉ Gioele Barabucci
gioele.barabucci@uni-koeln.de

[1] Cologne Center for eHumanities, Universität zu Köln, Köln, Germany

1 Introduction

Early in the process of setting up the digital environment for a critical edition of a work whose tradition is composed of witnesses in many different languages a problem arises: how to describe the relations between all these witnesses.

Classical textual scholars have a solid scholarly tradition to refer to when dealing with such an issue. Textbooks, theories and methodologies on how to write a stemma codicum abound. Much literature on how to deal with translated witnesses also exists, mostly thanks to biblical studies.

The literature, being written with classical scholars in mind, does not address in depth all the formal details needed to carry out a machine-based analysis of the evolution of text and its translations. Take, for example, the case of referring to part of a document. Different languages have different ways to divide documents and these have changed through the centuries. A simple reference, such as "the second paragraph", may refer to the wrong content in a certain translation or be completely nonsensical in the context of another translation written in an older language without the concept of paragraphs. For traditional textual scholars this is not a real problem: they know how to adjust their work to overcome these issues.

In digital scholarly editions, however, it is not possible to gloss over these details. Without a proper formalization, many of the tools of the digital scholar are no longer usable. To continue with the example of the references, think how useless a synoptic visualization tool like CATview (Pöckelmann et al. 2015; Barabucci 2016) would be, if it naively relied on paragraph indices to display the content of two documents written in two languages, each of which employs a different paragraph order.

The aim of this paper is to provide a set of formal and actionable definitions for many of the concepts that have to be addressed when creating a digital edition of works which have evolved over time (through authorial changes or copyist errors) and of which multiple translations exist (with some of them having a tradition of their own).

The content of this paper reflects the work being done in the context of the Averroes Digital Edition project at the Thomas-Institut of the University of Cologne.[1] The objective of the Averroes Digital Edition is to produce various scholarly editions of the works of the Andalusian Philosopher Averroes (also known as Ibn Rušd). The editions comprise Averroes's original Arabic texts as well as their Medieval Latin and Hebrew translations. To this end, all the known witnesses are being transcribed using a suitable TEI schema[2] and adding extra metadata about the origin of the text that they contain, for example, adding references to an ideal superstructure via a so-called chunking mechanism (Barabucci 2017). The fact that this extra metadata has to be encoded in formal terms in order to be read easily by machines has forced the project members to reflect on issues that are usually overlooked or, at least, not actively discussed and documented.

This paper provides, in Section 2, a discussion of the overall problem of describing the parallel evolution of a text and its translation. It then goes on in Section 3 to describe a series of possible approaches to this problem, starting with simple solutions to simple problems and progressively addressing more and more complex cases that require more advanced solutions. After this discussion from first principles, each fundamental concept

[1] http://averroes.uni-koeln.de/
[2] https://thomas-institut.github.io/averroes-tei

is isolated and formalized in Section 4. These formalizations have a double aim: providing a blueprint for an implementation and allowing a more precise discussion of these issues, including their inherent trade-offs. Future publications will describe how the abstract data structures described in this paper can be implemented in practice in standalone tools or as part of an environment dedicated to the study and production of digital editions.

2 The overall problem: How to describe the parallel evolution of a text and its translations

The problem of keeping track of how a document has evolved and how its translations have followed this evolution can been seen from two perspectives: from the present towards the future (*time-forward*) or from the present towards the past (*time-reversed*).

In the *time-forward* perspective we see the document evolution unfold: first a document is created, then its translations appear, then the document is modified and, subsequently, its translations are updated. This perspective is the perspective of the working translator. Understanding the evolution of the work is then quite easy: it is a matter of recording what has happened. The question is, however, "how, exactly?". A classical solution is a tree-like structure in which the chain of master documents forms the trunk of the tree and its translations are branches on the side. As the document evolves this tree grows with more and more branches.

In the *time-reversed* perspective we retrospectively rebuild this tree-like structure starting from the single items that will compose it and making educated guesses about their relations and the possible branches of the tree. This perspective is the perspective of the textual scholar. Part of the work of the textual scholar is to create a stemma codicum on the basis of plausible hypotheses. In other words, the textual scholar makes suppositions from the present about how the tree has evolved in the past.

In practice, these two perspectives are equivalent for what concerns the task of recording the relations between revisions of documents and their translations. The only difference lies in where the information about these relations comes from: in the time-forward case, the required information is recorded as time goes on; in the time-reversed case, the required information is provided by the scholar on the basis of their hypotheses.

Only the time-forward perspective will be used in this paper. Given that the data structures required to keep track of how the document and its translations have evolved are the same, it makes sense to use the perspective that is the most straightforward to work with.

One thing that will not be discussed is, surprisingly enough, the concept of *translation* itself. We will refer to Halverson (1997) for a thorough discussion on the different ways a document can be translated and on the various schools of thought on this topic.

3 Practical issues

This section deals with various practical issues related to the general problem of keeping track of the evolution of a document and its translations. In particular, we will investigate things like which pieces of information one has to record when a new revision of the master document appears, how to record that only part of a document

needs to be translated anew, and how to record that a translation has been compiled from multiple sources.

Discussing these issues allows us to incrementally build the set of requirements that our formalization will have to support.

3.1 The act of translation

In its most basic form, the translation problem comprises three parts:

- document A (the *master*), with content expressed in the language λ_A;
- document B (the *target*), the content of which is expressed in the language λ_B;
- the translator, that reads the content of A and produces the content of B.

As already stated, this paper will neither discuss the role of the translator nor the concept of translation. We assume that the translation act can be represented by a function *trans* that takes A, λ_A and λ_B as inputs and produces B. (This rough function definition will be refined and improved in the rest of this paper.)

At this stage things are quite simple: one file will store the content of A, another file will store the content for B. In this paper we will ignore details like which formats are used to store the content or where the content is stored. To keep things simple, we are going to assume that the content of A and B is stored in two distinct files that are readily accessible to the system.

3.2 The basic problem: Keeping master and target aligned

We start with a basic problem: the master document A has been modified and its translation B must be updated.

The fact that documents can be updated leads to the existence of multiple revisions of A and (eventually) of multiple revisions of B. To keep track of them, one needs a way to identify these revisions. In other words, one needs a revision ID. For the moment, we will identify the first revisions of A with A_α and A_β. Similarly we will use B_α and B_β to refer to the revisions of B that are translations of, respectively, A_α and A_β .

In addition, we can suppose that each revision is stored in a different file and that each file has a unique name. The names of the files are not directly related to the revision IDs.

We can visualize the evolution of a document and its translated versions as a graph, as shown in Fig. 1. The graph shows the evolution of A through time, and, the dependency between B and A.

In this simple case, to keep track of the evolution of A and its translation B we need a table with three columns:

- the revision ID, called υ,
- the path to the file where the content of revision A_υ is stored,
- the path to the file where the content of revision B_υ is stored.

An empty value in the third column signals that a certain revision has not been translated yet. In the generic case of N different target languages, the table will have

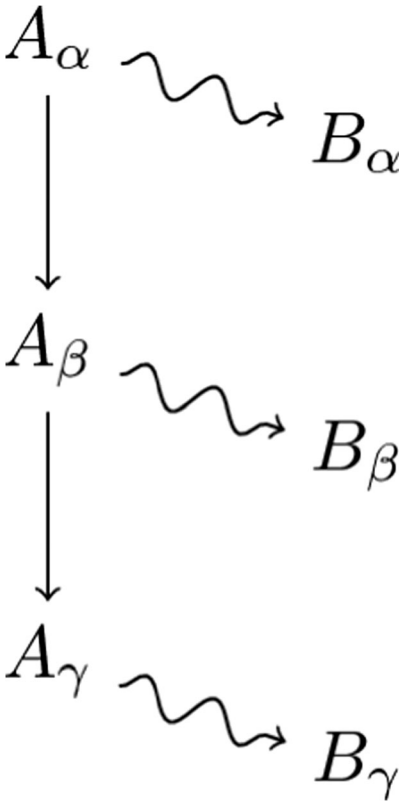


Fig. 1 Evolution of A and of its translation B. A straight line denote an authorial change leading to a new revision, a weavy line indicates a translation leading to a new document

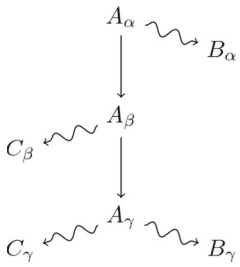
2 + N columns. Figure 2 shows a graph with the evolution of a document and its translations, together with the equivalent revision table.

Each modification to the master document introduces a new row in the revision table. Initially, a new row contains only the revision ID and the path to the content of the master document; the cells referring to the translations are empty. As this revision is translated into other languages, the respective cells are filled with the path to the translated content.

This simple revision alignment table is useful to keep track of documents that have a clear evolution path and whose translations are orderly produced from the master document. In reality, though, this scenario is quite rare and more advanced techniques are required to keep track of how a document and its translations have evolved.

3.3 Translations of translations

There is a common translation practice that the simple revision alignment table shown in the previous section cannot deal with: the translation of a document from another translation. This happens, for example, when the document C is translated from another translation B instead of using the master document A.



(a) *Graph form*

υ	A	B	C
α	A1.txt	B-18-jan-03.txt	—
β	A2.txt	—	C-72d3.txt
γ	A3.txt	B-18-feb-12.txt	C-2ac30.txt

(b) *Tabular form*

Fig. 2 Evolution of A and its translations B, C

While, in theory, the document C is a translation of A, in practice, going though the intermediary translation B may introduce translation errors. It may well happen that B is a valid translation of A and that C is a valid translation of B, but, in fact, C is not a valid translation of A.

In mathematical terms, the root problem stems from the fact that the translation function *transl* is not a transitive function.

Therefore, the use of an intermediary translation needs to be recorded and traced.

This extra piece of information complicates our tracking system: we can still draw the graph as a tree, but the revision table must be split. Each language will have its own three-column table that associates the revision ID with the file path and with the origin, i.e. the document from which a certain revision has been translated. The updated graph is shown in Fig. 3, the split tables in Fig. 4.

3.4 Contaminations

An even more complicated case is when a translation has been derived from other translations or from multiple sources, i.e. some kind of contamination has occurred. An example of this is the way in which the translation teams in various EU legal departments work (Cavoski 2017; Schäffner 2001).

The use of multiple sources is, upon closer inspection, something that is intrinsic to the way translations are updated. When a new revision A_β is released, its translation B_β is often produced using the existing translation B_α in addition to A_β itself.

Tracking the fact that a translation may have more than one source has two effects.

First, we will have to modify the trans function to accept more than one source document and one source language. In principle, an arbitrary large number of documents could be recorded as being the source for a certain translated revision.

Second, the derivation tree is no longer a tree, but an acyclic direct graph (cycles are not possible, because a future document cannot be the source of a past translation). This change also reflects on the derivation table that now allows for multiple source documents to be recorded. An example of the new graph and of the new table are show, respectively, in Fig. 5 and Fig. 6.

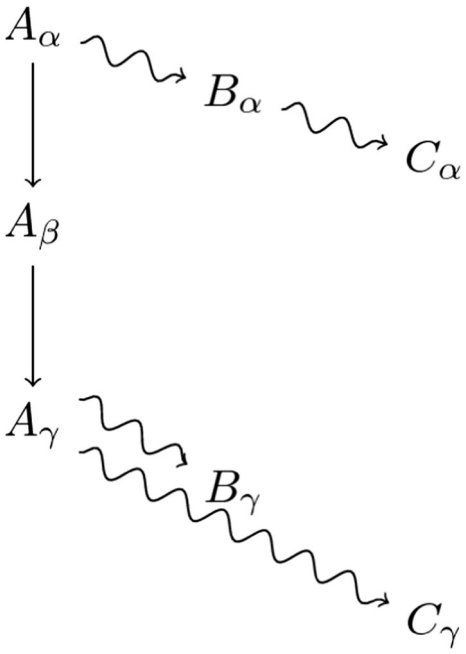


Fig. 3 An evolutionary tree with intermediary translations

3.5 Updates to part of a document and partial translations

Up to now, we have considered documents as atomic documents: a whole document is updated, a whole document is translated, a whole document is used a source for a translation. This is an extremely simplified view of the reality.

Under this model, updating A produces A_β. In turn, B_β is produced by translating A_β from scratch, ignoring the work done to translate the previous revision B_α.

In practice, however, what is commonly done is translating only the parts of A_β that have been changed since A_α. This approach can drastically reduce the amount of work done to update B.

Translating only the changed parts of A introduces, however, two new problems:

- How are we going to divide A in parts?
- How do we know that certain parts of A have been changed?

υ	A	orig
α	…	—
β	…	—
γ	…	—

υ	B	orig
α	…	A_α
β	—	—
γ	…	A_γ

υ	C	orig
α	…	B_α
β	—	—
γ	…	A_γ

Fig. 4 Tabular representation of the tree in Fig. 3 with intermediary translations (all file paths replaced by …)

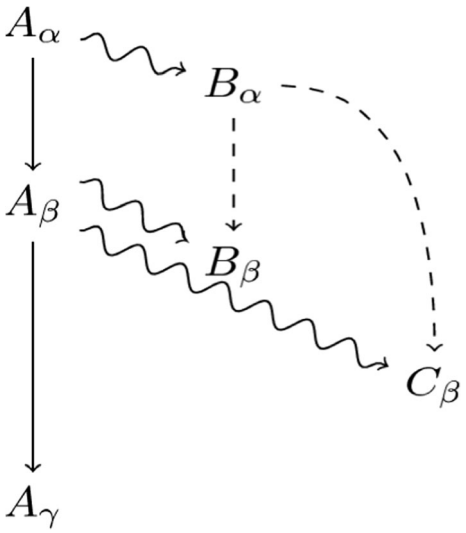


Fig. 5 An evolution graph with translations derived from multiple sources. A dashed line denotes the use of a secondary source, e.g. the translation of a previous version or an intermediary translation

The first issue, dividing a document into parts, is both a practical problem and a highly theoretical problem. In general, the parts into which a document can be divided depends on the model imposed (or that can be imposed) on the content in that model. For example, an essay may be divided in paragraphs, a poem in lines often grouped in stanzas. In particular, electronic documents can be divided in a plethora of different ways, related to the many different levels of abstractions that can be recognized in a document (Barabucci 2019).

Associated with a way to divide a document in multiple parts, there must also be an addressing scheme to refer to these parts. For the moment, we will suppose that documents are nothing more than a sequence of paragraphs. In other words, they can be divided in paragraphs and each paragraph in a document has an associated index; for example $A_\alpha/2$ refers to the second paragraph in A_α.

The second issue, identifying which parts of A_α and A_β are different is what is known as the *diff problem* (Barabucci 2013). A basic *diff function* returns the indexes of the parts of A_β that are different from their counterparts in A_α. For example, if A_α was

υ	A	orig
α	…	—
β	…	—
γ	…	—

υ	B	orig
α	…	A_α
β	…	A_β, B_α
γ	—	—

υ	C	orig
α	—	—
β	…	A_β, B_α
γ	—	—

Fig. 6 Tabular representation of the graph in Fig. 5 with translations derived from multiple sources (all file paths replaced with …)

composed of the three paragraphs "X Y Z" and A_β of "X K W", the diff function would return (2,3), i.e. it would state that the second and third paragraphs are different. In practice, even the most basic version of the output of a proper diff function has to be much more complicated than that because it has to take into account not only in-place modifications but also, as a minimum, the insertion of new paragraphs and the deletion of existing paragraphs. For instance, if A_α were "X Y Z" and A_β were "X K Y W", the diff function will state that K has been added after X, that W has been added after Y, and that Z has been removed. For the sake of brevity, we will assume that an appropriate diff function exists and we will ignore the problem of *how* the diff function detects differences between revisions of a document.

In practice, the output of a diff function can be seen as an alignment table with two columns: the ID of a paragraph in A_α and the ID of the corresponding paragraph in A_β. The first column will be empty for paragraphs added in A_β, the second column will be empty for paragraphs deleted from A_α. Such an alignment table is shown in Fig. 7.

Once one knows which parts of A_β differ from their respective parts in A_α, one can understand which parts of B_α can be reused and which parts have to be translated from A_β.

It must be noted that this alignment table is unrelated to the previous revision table. It is just a means to understand what is "new" and has to be translated. The alignment table could be thrown away once the target documents have been updated.

What about our evolutionary tree and the associated revision tables? Do we need to extend them to take into account that documents may be divided in parts, for example in paragraphs? If we assume that the number and the order of the paragraphs is the same in the master document and in each target translation, we do not need to change anything.

3.6 Different scripts, grammars and subdivision styles

To assume that the number and the order of the paragraphs or sentences is going to be the same in all target languages is, however, unrealistic, especially when working with

A_α	A_β	
1	1	X (in both documents)
—	2	K has been added after X
2	3	Y (in both documents)
—	4	W has been added after Y
3	—	Z has been removed

Fig. 7 Alignment table and differences between A_α (X Y Z) and A_β (X K Y W)

languages that have quite different grammars and rhetorical constructions, for example English and Chinese, or Latin and Arabic.

For example, Arabic in the twelfth century was written with no punctuation and there was no clear distinction between paragraphs or sentences. Latin, on the contrary, had already abandoned *scripto continua* and the division in paragraphs was being introduced via rubrics (Saenger 1997). This means that an Arabic text being translated into Latin was going to be divided into parts by the translator, with the boundaries of these parts being quite arbitrary and with the possibility of parts of the text being transposed for the sake of style.[3]

There is the need, thus, to track which parts of the document A has been used as the source of which parts of the target document B.

To keep track of this information over the whole life of A and B we need a new source-part table, in addition to the tables that we have seen up to now. In this alignment table we are going to record which parts of A_υ are the source for which parts of B_υ.

The new source-part table has three columns:

- the ID υ of the document version,
- the ID of the part in A_υ,
- the ID of the part in B_υ.

An example of such a source-part alignment table is shown in Fig. 8.

3.7 Revisions of translations

So far we have assumed that the master document gets updated but translations do not. However, translations do get updated. Some translations even start a tradition of their own.

If we accept the idea that a translation of a certain revision can have revisions of its own, then we need to keep track not only of which revisions of B are translations of which revisions of A, but we also need to keep track of how the various revisions of B relate to one another.

To store these new pieces of data, we need to modify the structure our tables. A simple list of revisions like we have hitherto used is no longer enough. We must give a unique name to each revision of each document in our graph, like in Fig. 9, and then store the association between them in a new table, such as that shown in Fig. 10.

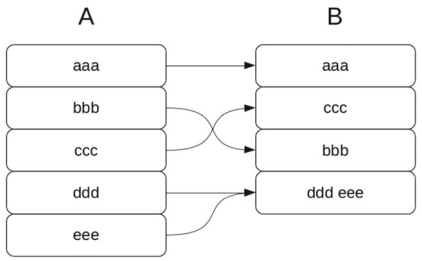
4 Formalization of the fundamental concepts

In the previous section we have seen a variety of problems that are routinely faced by the scholars that want to track the evolution of a document and its translations. These problems were presented in a discursive manner and in an incremental way.

In this section, instead, the focus will be on formalizing the separate fundamental concepts that are behind the problem of tracking the parallel evolution of documents.

This formalization has a double aim. On the one hand, this formalization represents an actionable guide for the implementation of a system that can be used to track the

[3] For practical examples, see the chunking system used by the Averroes project.



(a) *Graphical representation of the relation between the content of A and B*

υ	A	B
α	1	1
α	2	3
α	3	2
α	4	4
α	5	4

(b) *Alignment between parts of A and B*

Fig. 8 Source-part table

evolution of documents and translations. On the other hand, such a formalization opens the door to a more precise discussion about these concepts.

4.1 Documents according to CMV+P

The first concept one must formalize is that of the document. There are many definitions of what a (digital or non-digital) document is. The function of these definitions is to impose a particular view of what a document should be, which kind of data it should contain and so on. In this paper we will make use of the so-called *Content, Model, Variants + Physical level* (CMV+P) document model (Barabucci 2019).

In its most basic form, the linear version, the CMV+P model describes each document as a stack of abstraction levels. Each level in the CMV+P stack is composed

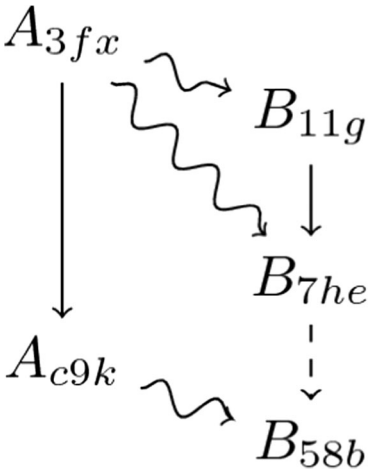


Fig. 9 An evolution graph with multiple revisions of a translation. As in the previous figures, a straight line denotes an authorial change leading to a new revision, a weavy line indicates a translation leading to a new document, and a dashed line symbolizes the use of a secondary source in the creation of certain document

υ_A	υ_B
3fx	11g, 7he
c9k	58b

υ	A	orig
3fx	…	—
c9k	…	—

υ	B	orig
11g	…	A_{3fx}
7he	…	A_{3fx}, B_{11g}
58b	…	A_{c9k}, B_{7he}

Fig. 10 Tabular representation of the graph in Fig. 9 with a translation undergoing multiple revisions

of a) an addressable *Content*, b) a *Model* according to which the content has been recorded, and c) a set of *Variants* used for equivalence matching. At the bottom of this stack is the *Physical* level, that symbolizes the concrete medium in which the document is embodied. These abstraction levels are connected by transformation functions that map the content of the upper abstraction level to (encoding) and from (decoding) the content of the lower abstraction level.

Examples of stacked abstraction levels are the TEI and the XML abstraction levels. At the TEI abstraction level, the Model is the TEI specification and the Content is a tree of titles, sections, paragraphs, figures and so on. At the lower abstraction level XML, the Model is the XML specification and the Content is a tree of nodes, elements, attributes and comments. These two levels are connected by transformation functions that map the content of the upper abstraction level to and from the content of the lower abstraction level.

4.2 Translation function

Translated documents are the results of a translation function that we will name *transl*. The *transl* function represents the act of translating a document.

At the document level, this translations function takes in input:

- one master document, written in certain language,
- zero or more secondary documents, each written in a certain language,
- a target language.

These parameters are necessary in order to perform the translation itself.

Formally, the translation function at the documentation level can be written as

$$transl : Doc_a \times lang_a \times (Doc_i \times lang_i)^n \times lang_{target} \rightarrow Doc_{target}$$

Where *Doc* is a document at a certain abstraction level, $lang_x$is the language of the document x and n is the number of secondary documents used in the translation.

For example, the fact that the document B_β is a translation of A_β and has been created reusing parts of B_α would be recorded as

$$transl\left(A_\beta, \lambda_A, B_\beta, \lambda_B, \lambda_B\right) = B_\beta$$

The *transl* functions is not a transitive function. That means that if C has been translated from B and B has been translated from A, it may be the case that C is *not* a proper translation of A.

More formally, the fact that

$$transl(A, \lambda_A, \lambda_B) = B, transl(B, \lambda_B, \lambda_C) = C$$

does not imply that

$$transl(A, \lambda_A, \lambda_C) = C$$

One implication of the non transitivity of the *transl* function is that each intermediary translation must be recorded.

4.3 Equality/equivalence between documents

When can we say that two documents are different? When are two aligned parts in two revisions of a document different?

Generically speaking, the answer to these question is application- and model-specific: it always depends on the point of view of the person asking and on the model used to describe the documents. For example, in certain cases, capitalizing a word may be seen as a trivial change that does not make the document really different, while in other cases such a change may be regarded as critical.

The CMV+P document model provides a formal framework to answer these questions. In particular, it provides a formal definition of equality and equivalence.

According to CMV+P, two documents are equal at a certain level of abstraction when their respective C sets (Content sets) for that level of abstraction are identical: they contain the same number of elements, in the same order and all elements are pairwise equal.

Conversely, two documents are equivalent at a certain level of abstraction when some of the elements in their C sets are different, but all these different elements have an associated record in their V set (set of Variants) and there is an equivalence between these records.

These formal definitions do not solve the problem of understanding if two documents are different and, if so, how. These definitions provide, however a formal framework to describe, for example, which phenomena can be considered variants.

4.4 Revision identifiers

Revisions must be uniquely identified, regardless of how one defines what constitutes a new revision. A system of identifiers must be in place to give each of these revisions a referenceable name.

The simplest way to identify revisions is to associate a global progressive identifier to each one. This means that A_1 is the first revision of A, A_2 is the second and so on. At the same time, B_1 is the translation of A_1, B_2 of A_2 and, in general, B_n is the translations

of A_n. This technique has a shortcoming: it can accommodate only cases where each translation has exactly one revision.

Another way to generate revision identifiers is to generate them from the content of the document using a mapping function *revid* that takes a Content set and returns an unique ID. Each ID will then identify with precision only a precise arrangement of the content.

$$revid : C \rightarrow ID$$

The content used to generate these identifiers must be understood as the Content set at a certain abstraction level. Which exact abstraction level should be used is an application-specific choice.

In simple cases, e.g. when the abstraction level is a bitstream level, the *revid* function can be implemented via an hash function such as SHA1.

4.5 Division in parts and indexing

The model of a document is what decides how a document can be divided. Bibles, for example, are divided in books and subdivided in chapters and verses; poems on the other hand are divided in stanzas and verses.

The fact that the use of different models lead to different kinds of elements is reflected in the CMV+P model, where the kinds of elements present in the Content set at a certain abstraction level are dictated by the Model of that level.

An indexing mechanism—i.e. a way to name these different parts—is needed if we want to keep track of the influence of these parts on other documents or if we want to state that they have been modified in a certain revision.

There are three kinds of indexing mechanisms:

- sequential indexes, for documents whose parts have an implicit order (for example, the paragraphs in a letter);
- path indexes, for tree-like documents with nested structures (for instance books that are divided in chapters, subchapters, paragraphs and so on);
- direct indexes (or labels), for graph-like documents (hardly found in textual studies).

4.6 Derivation graph

The derivation graph is the data structure that stores the information about the relations between the revisions of the master document and its translations.

It is described via three kinds of tables: the revisions-alignment table, the source-document table and the source-part table.

4.6.1 Revision-alignment

The revision-alignment table describes which revision IDs are to be considered equivalent, i.e. all the revisions on the same row are all translations of the same content.

There is only one revision-alignment table per derivation graph.

This table has one column for each language and one row for each revision of the master document. Each cell contains a list of revisions IDs.

v_A	v_B	…	v_C
3fx	11g, 7he	…	93i
c9k	58b	…	qa4

4.6.2 Source-document table

The source-document table describes two things:

- where the content for a certain revision of a certain document is, and
- what are the other documents that have been used to translate this revision.

There is one source-document table for each language.

This table has three columns:

- the revision id,
- the path to the stored content for that revision in that language,
- the list of documents that have been used to translate this content

Each row of this table represents a revision in a certain language.

v	B	orig
11 g	B-18-jan-03.txt	A_{3fx}
7he	B-20-jan-03.txt	A_{3fx}, B_{11g}
58b	B2-08-sept-04.txt	A_{c9k}, B_{7he}

4.6.3 Source-part table

The source-part table describes the alignment between the parts of two documents at a certain abstraction level.

There is one source-part table for each pair of languages.

This table has four columns:

- the revision id of the master language,
- the revision id of the target language,
- the index of a part in the master language,
- the index of the same part in the target language.

Each row in this table represents the alignment between a part of the document in the master language and a part of the document in the target language.

v_A	v_B	A	B
7w1	xk4	1	1
7w1	xk4	2	2
7w1	xk4	3	2
c9k	58b	1	1
c9k	58b	2	2
c9k	58b	3	3

5 Conclusions

This paper presented a series of formalized data structures that can be used to keep track of how a document and its translations have evolved through time.

These data structures provide the foundation for a revision tracking system that understands the concept of translation and that can be used in a digital environment to support the work of textual scholars. One practical use of these data structures is to encode the stemma codicum of literary works whose tradition is spread over different languages.

In the future, extensions to the shown data structures will be presented, in order to address traditions where there is no concept of "master document" and to keep track of the so called "overlay locations", i.e. voluntary aberrations from a normal translation that are preserved even when the master is updated.

References

Barabucci, G. (2013). A universal delta model. PhD thesis. Università di Bologna. https://doi.org/10.6092/unibo/amsdottorato/5761.

Barabucci, G. (2016). CATview (review). *Digital Medievalist*, 10. https://doi.org/10.16995/dm.57 .

Barabucci, G. (2017). *Not a single bit in common: Issues in collating digital transcriptions of Ibn Rusd's writings in multiple languages (Arabic, Hebrew and Latin)*. Presented at Digital Humanities Abu Dhabi 2017 Conference. New York University Abu Dhabi.

Barabucci, G. (2019). The CMV+P document model, linear version. In R. Bleier and V. Das Gupta (Eds.), Versioning cultural objects. IDE. (in print).

Cavoski, A. (2017). Interaction of law and language in the EU: Challenges of translating in multilingual environment. *Journal of Specialised Translation, 27*, 58–74.

Halverson, S. L. (1997). The concept of equivalence in translation studies: Much ado about something. *Target: International Journal of Translation Studies, 9*(2), 207–233. https://doi.org/10.1075/target.9.2.02hal.

Pöckelmann, M., Medek, A., Molitor, P., & Ritter, J. (2015). CATview: Supporting the investigation of text genesis of large manuscripts by an overall interactive visualization tool. Presented at Digital Humanities, DH2015, Sydney.

Saenger, P. (1997). Space between words: The origins of silent reading. Stanford University Press: Stanford. ISBN: 9780804740166.

Schäffner, C. (2001). Translation and the EU: Conditions and consequences. *Perspectives: Studies in Translation Theory and Practice, 9*(4), 247–261. https://doi.org/10.1080/0907676X.2001.9961422.