Intertextual Pointers in the Text Alignment Network

Joel Kalvesmaki

Journal of Data Mining and Digital Humanities

Intertextual Pointers in the Text Alignment Network

Joel Kalvesmaki[1]

1 Dumbarton Oaks

*Corresponding author: Joel Kalvesmaki kalvesmaki@gmail.com

Abstract
The Text Alignment Network (TAN) is a suite of XML encoding formats intended to serve anyone who wishes to encode, exchange, and study multiple versions of texts (e.g., translations, paraphrases), and annotations on those texts (e.g., quotations, word-for-word correspondences). This article focuses on TAN's innovative intertextual pointers, which, I argue, provide an unprecedented level of readability, interoperability, and semantic context. Because TAN is a new, experimental format, this article provides a brief introduction to the format and concludes with comments on progress and future prospects.

keywords
XML, TAN, TEI, pointers, alignment, intertextuality, canonical references

I XML and Intertextuality

For the scholarly study of texts, encoding in XML has been critical. The syntax is simple, and better suited to encoding texts than are, say, JSON or relational databases. The maturation of ancillary XML technologies such as XPath, XSLT, XQuery, RELAX-NG, XSpec, Schematron, and Schematron Quick Fixes have allowed efficient querying, validation, testing, and transformation of XML files.

Those technologies have allowed groups to define specialized uses of XML, and use those defined schemas to drive important research. In the case of text markup, many of us owe a debt of gratitude to the Text Encoding Initiative (TEI). TEI is a customizable suite of schemas that define an enormous library of elements and attributes (567 and 258, respectively, in the largest version of the schema, TEI All). The library models a great number of textual feature types that scholars are likely to want to use to annotate their texts. Although TEI is talked about as being a standard, it does not fit the way most people understand the term « standard ». It was designed with a laissez-faire view of the markup task, and so supports an enormous range of research assumptions and questions, and it does not broker legitimate differences of opinion on how a text should be analyzed. The TEI Guidelines suggest, but do not enforce, the use and interpretation of a given element or attribute, and the Guidelines admit ambiguity and vagueness—necessarily so, given the decision to make it as universal as possible.

Because of this design, TEI is not well suited to cross-project interoperability (Schmidt 2010, Schmidt 2014). The TEI guidelines require interpretation, and even scholars who agree on how to interpret them may legitimately mark the same textual features in different ways. Such pluralism means that if you wish to incorporate someone else's TEI file into your project, you must first study it, then write a conversion that suits the way you have interpreted and used TEI. The exercise must be repeated for every imported file, and the work that goes into the conversion ends up benefitting only your project. Everyone else must repeat the same task on their own.

Interoperability of multiple versions within a single project is relatively easy. Many TEI projects have done just that. But TEI offers no mechanism to try to check or enforce cross-project alignment. Four different projects that each encode a version of Aristotle's *Categories* are likely to come up with as many ways of structuring the transcriptions in TEI. Anyone who wishes to synthesize those four versions, say into an HTML page that aligns the versions in parallel, must spend considerable effort reconciling differences. And every time a new TEI version of the *Categories* is found, the exercise must be repeated.

Or consider the humble cross-reference. TEI allows many ways to represent a claim that work A passage x is a quotation of work B passage y. But any representation in TEI is almost guaranteed to be meaningless outside a given TEI file or project (Kalvesmaki 2015). TEI code requires both syntactic and semantic interpretation, and is not predictably applicable to other versions of the same work.

I've singled out TEI, but the problem is true for other markup schemes. HTML, TMX, and XLIFF allow users to align multiple versions of the same work, but no effort has been made to try to make data interoperable across projects using the same markup method.[1]

II A New Approach to Intertexts: The Text Alignment Network

I have often wondered what would have happened if the designers of TEI had taken a different approach. Could TEI be constrained to maximize the likelihood that the texts I produce, and the things I want to say about them, will be interoperable with other versions and annotations you produce? Can I write intertextual pointers that have value not just for my TEI transcription, but to the ones other people have made for the same work?

Over the last several years I have worked to answer "yes" to both questions by means of the Text Alignment Network (TAN), a suite of XML encoding formats intended to serve anyone who wishes to encode, exchange, and study translations, paraphrases, adaptations, quotations, and other varieties of text reuse. The TAN syntax is meant to be maximally readable and editable by both humans and machines. The RELAX-NG and Schematron schemas that drive the validation routines not only check the syntactic and semantic interoperability of a text, but provide feedback and help (via Schematron Quick Fixes) in the context of an XML editor such as oXygen.[2]

TAN is a format, not a tool. It does not make intertextual pointers for you. It does not try to identify quotations or to reconcile differences in alignment. But it lays a foundation for such efforts. The extensive library of XSLT functions upon which TAN validation is built definitively interpret the format and provide a starting point for further use (some examples are discussed at the end of this article).

TAN is not intended to replace other formats such as TEI. In fact, TAN is built partly on a customization of TEI All. Rather, TAN is meant to complement other formats, because it has been designed around a set of distinctive principles:

- **Annotations should always be separated from what is annotated (stand-off markup):** Anything that is not a transcription but comments on one—e.g., remarks on quotations, the analysis of the morphological parts of a given word—should stand apart from the transcription itself. Such stand-off annotation brings many benefits not available with inline annotation. Multiple scholars can edit and remark on the same transcription independently and collaboratively. Any two annotations may point to overlapping ranges—one of the Great Impermissibles in a single XML file—without compromising XML rules.[3] Further, stand-off

[1] For TMX and XLIFF see https://en.wikipedia.org/wiki/Translation_Memory_eXchange and https://en.wikipedia.org/wiki/XLIFF.

[2] https://www.oxygenxml.com/ (accessed 2017-10-17).

[3] Other alternatives to the Great Impermissible have been proposed, e.g., the Layered Markup and Annotation Language (LMNL). http://lmnl-markup.org/. Such efforts dispense with XML altogether, and so cannot

ISSN 2416-5999, an open-access journal

files can be combined and transformed into any number of derivative inline files. The reverse is not always true. A heavily marked up TEI file is like a baked cake. Each type of tag set is like a set of ingredients or form of preparation. You might get a TEI file shaped baked into the shape of a star, sprinkled with walnuts (I'm allergic to them). But to make it the cake *you* want, you have to unbake it, remove the ingredients you don't like, add the ones you do, then bake it again. In adopting stand-off annotation I argue that we should not be baking cakes. Rather we should be providing simple ingredients, so others can bake cakes without having to first unbake them.

- **One file per task** : As a corollary to the previous principle, a given file should be focused upon a single task, and kept simple. Each version of a work should be in its own transcription file, and restricted more or less to the plain text. An index of quotations should be in a separate file, and simply point to the transcription files rather than trying to mark them up. Most important is the requirement that every TAN/TEI transcription file must be restricted to a single version of a single conceptual work found on a single text-bearing object, segmented and labeled according to a single reference system (canonical or not). If we are working with a book that features multiple versions of a work, each version of interest needs to be broken out into its own file. The purpose of a TAN/TEI file is to focus on the work, not on the physical exemplar, which means that a TAN transcription file is likely to cover only a portion of what appears in a book.
- **Metadata should be focused on the task at hand :** But because a given file is devoted to one task, the metadata should as well. Yes, we want to know who was responsible for editing or proofreading a text. But we don't need to know the proofreaders' age, nationality, or background. A unique identifier for those people can point us to other places more authoritative where we can learn about them. So too with a book that is the basis for a transcription. We don't need a full catalogue entry. Our data is not about the book, it's about the work. Just point to one or more records where we can learn more, if we so choose. Metadata should be restricted to the data in the <body>.
- **The bare scholarly minimum should be expected and enforced in metadata :** The metadata in a TAN file should fulfill scholarly expectations. It must *de minimiis* answer a core set of questions scholars need to know to be able to use the data responsibly: what is the name and version of this file, what are the sources of the data, under what license is the data released, who created the data and when, and what assumptions and definitions were adopted in editing the data. Neither empty fields nor obviously fake responses should be permitted.
- **All data must be human- and computer-readable :** Everything in a TAN file must be both human- and computer-readable. The computer-readable component is oftentimes an Internationalized Resource Identifier (IRI, roughly synonymous with URI, Uniform Resource Identifier, i.e., a URN or URL),[4] so that TAN files can contribute to the Semantic Web.
- **Content and assertions should be deeply validated :** It is fine, of course, that our files' structure is checked for validity, but the content should be checked as well, when possible. If a word is marked as having particular grammatical features, the code we use should comport with the rules we have adopted for the language (e.g., that there should be no such thing as a first-person nominative conjunction). Text should be checked to make sure it is normalized (according to Unicode NFC). In a given element with both @from and @to, the date in the latter should not precede the one in the former.

  In TAN, more than one hundred such rules are checked, at different depths. Users are permitted to specify how deeply these checks should be made, to save time during editing. The underlying XSLT functions have been designed to provide contextual help to editors through Schematron Quick Fixes (SQFs). If an SQF-aware XML application (e.g., oXygen) flags text

immediately take advantage of related XML technologies (e.g., XPath) without first being converted to derivative XML documents.

[4] The TAN guidelines prefers the term IRI, because IRIs accommodate most of Unicode. But this distinction is subtle and minor. All URLs and URIs are IRIs, and all IRIs can be expressed as a URI.

ISSN 2416-5999, an open-access journal

as not normalized, the software will give the user the option to automatically replace the text with a normalized version, saving perhaps hours of hunting and retyping.

- **We should be able not just to collaborate but to communicate across projects** : As textual scholars, we should be able to work not just independently but collaboratively. Good collaboration should include some form of communication. Suppose you have published on your website a TAN transcription that I find important and I want to refer to it in my own TAN file. To do so, I must specify when I last consulted your file (via @accessed-when). Now, suppose you find errors in your file. You make the correction, and explain and date this change in your file. The next time I validate my dependent TAN file, I will be informed or warned that your source file has since changed. Suppose your alteration is serious. You can specify how serious it is, and describe the correction. My validation process will not only mark my file at the level of error you've specified but convey your message. I can ignore that message if I want, or I can take into account the changes you have made. If I update @accessed-when, I can squelch those older messages.

Much more could be said about the design principles, but such an effort would wind up repeating the comprehensive documentation already available at the project's website, http://textalign.net. All material is released under a Creative Commons Attribution 4.0 license, in the hopes that the project will outlive and outgrow me.

III Pointers

1 Current Pointing Systems

I wish here to focus on TAN pointers, since they are at the heart of intertextuality, the theme of this issue of the journal. TAN pointers are most similar to those defined by Canonical Text Services. But, or so I argue, they provide a new level of readability and refinement. TAN pointers are provide an unprecedented level of integration with the semantics of the source and the target. And they show that the syntax we already use in the notes of our articles and books to quote primary sources is adequate for digital texts. We don't need long, cumbersome URNs ; we can use shorthand forms that are already familiar.

For comparison it is worthwhile mentioning some other pointing mechanisms that are well known and have been around for a long time:

- HTML : click here
- XPointer : <xi:include href="http://example.com/index.htm " xpointer="chapter1"/>
- XSLT: <xsl:copy-of select="//tei:div[@type = 'chapter' and @n='1']/tei:p" />
- TEI (one method): <link target="#chapter1 #chapter2"/>
- TEI (another method) : <app corresp="http://example.com/example.xml##range(left(//lb[@n='3']),left(//lb[@n='4']))"/> (Cayless 2013, ¶18)

A few observations are in order, treating readability, interoperability, and semantics.

First, each of these pointers have limited readability to any human who needs either to write or to read them. Different people respond differently to these kinds of pointers, based on familiarity. The URLs (HTML and XPointer) and IDRefs (HTML, XPointer, TEI) might be recognizable, even if long. The pointers dependent upon XPath (TEI) will probably not be clear to those who do not work in XSLT or XQuery. All of them, though, are pretty long and cumbersome. Few of us type them out ; we normally rely upon cutting and pasting.

Second, each pointer has coinage in only one file. The HTML, XPointer, and TEI examples point to a single node within a single document. Even if they could be tried on other files (like the XSLT example, which is meant to be applied to an unspecified number of files), the intention built into the pointer will be respected only if those other files have been encoded using the same syntactical markup. But the syntax is necessarily arbitrary, and people

working independently will come up with their own naming systems. So any success will be a happy but rare coincidence.

Third, the underlying semantics are ignored. Even if we humans intuit their meaning, the terms « chapter1 » and « chapter » are not computationally defined. The syntax may be computable, but the computer has no means to fetch the meaning (semantics) behind that syntax. That is, a computer does not « understand » (we seem forced to anthropomorphize the algorithms we write) that « chapter » refers to a such-and-such a type of textual division.

The examples above illustrate simple, one way pointers. Now, consider the humble cross-reference, which requires at minimum a double pointer, set in the context of a sentence. Suppose we want to encode the idea « Matthew 4:15-16 is a quotation of Isaiah 8:23-9:1. » (Example A.) And suppose we want to use one of the traditional pointers above. The first issue, readability, still holds, although its greatest sin is its nuisance. The second issue, interoperability, presents more serious obstacles, since the sentence we want to requires us to use a syntax that transcends any individual version of the work. But all the pointing techniques above are restricted to individual files. And the third issue still holds : none of the pointing mechanisms are able to preserve the semantics behind the references, e.g., that the grammatical subject refers to a book of the New Testament.

Suppose we wanted to be more specific in our assertion. After all, there are parts of Isaiah 8 :23 that are not part of the quotation, which doesn't begin until around the 15th word, according to the Rahlfs edition of the Septuagint. Furthermore, a number of words or phrases differ between the quoting and quoted source. Revising Example A to be more specific could get pretty complicated. For the sake of illustration, we'll stay simple : « All of Matthew 4 :15-16 is a quotation of Isaiah 8 :23 word 15 through the end of Isaiah 9 :1. » (Example B.)

Strangely, the traditional pointing methods above are better suited to Example B than they are to Example A. That is because the latter was meant to be true independent of any version of the texts. We were flying about in a sort of zero gravity. But now that we are seeking to point to individual words, we must enter the gravitational field of a specific version, at least for the second text (and perhaps the first). We cannot expect a statement true of the Rahlfs edition of the Septuagint to be true for the Brenton one (where the quotation doesn't begin until the sixth word of Isaiah 9 :1 !). Which means that our second concern above, about interoperability, is not as relevant, since we must restrict our assertion to specific versions, which is what the pointing languages above were designed for. (But all the other concerns still hold, and even the issue of interoperability of multiple files handling the same version.)

I have framed the problem in this way because it seems to me that all textual pointers fall into one of two categories. A pointer either transcends any particular version or it does not.

Pointers from the first category rely upon a reference system that must necessarily be version-agnostic, even if that reference system originates from features endemic to a specific version. For example, the page numbers, column letters, and line numbers of Bekker's edition have been used as a standard reference scheme for Aristotle's works, applied to later print editions. But the typesetting in those later editions results in lineation that differs considerably from Bekker's. Typesetters have not tried to match exactly Bekker's line beginnings and ends, so a Bekker number is oftentimes only a good approximation. The Bekker reference system has transcended Bekker's edition.

In the second category are pointers that delve into a particlar version. The pointer may begin with a version-agnostic component (e.g., Isaiah 8 :23…) but they must terminate in something that does not (…word fifteen). Or it the pointer may dispense with the version-agnostic reference altogether, and rely completely upon a reference system that is native to

only a single version (e.g., quire 43, fol. 4r, col. 4, line 43, character 11, which is where Isaiah 8 :23 begins in Codex Sinaiticus).

2 Version-agnostic pointers

Let us start with version-agnostic pointers such as those in Example A. Elsewhere in this special issue we have read about Canonical Text Services URNs,[5] which has been to my knowledge the first attempt to provide a digital language for pointers that transcend any specific version, and so strive to admit syntactic interoperability. So for Example A, we would begin with CTS URNs such as this (adopting the somewhat problematic catalog numbering of the Thesaurus Linguae Graecae [TLG]):[6]

- urn:cts:greekLit:tlg0031.tlg001:4.15-4.16
- urn:cts:greekLit:tlg0527.tlg048:8.23-9.1

These URNs reliably point to Matthew 4:15-16 and Isaiah 8:23-9:1 independent of any mechanism, tool, or server.[7] They point to whatever version you can find of the cited works. But they are still just as cumbersome, opaque, and unreadable as the traditional pointers discussed above. And although they are syntactically interoperable, they are not semantically so (Kalvesmaki 2015). Nevertheless, it is admirable that we can pack so much into a single URN. A CTS URN is a good start to expressing Example A. To finish the job, we might adopt the data model of the Resource Description Framework (RDF), which defines three-part statements, made of subject, predicate, and object. Here is one way of using CTS URNs to create an RDF statement for Example A (using the Turtle syntax, and adopting a URL that has been minted to define the concept « quotation ») :

```
urn:cts:greekLit:tlg0031.tlg001:4.15-4.16        http://purl.org/spar/fabio/Quotation
urn:cts:greekLit:tlg0527.tlg048:8.23-9.1 .
```

The TAN approach also uses URNs (IRIs), but in the interest of readability, those IRIs have been moved into the background. They are retrieved either by looking up the value of an id value, or by calculating the value of a node from its context. Splitting the work this way allows editors to spend most of their time working with a syntax that close to ordinary conventions. Here is how Example A could be rendered in a TAN annotation file:

```
<claim verb="quotes">
  <subject work="nt-grc" ref="Mt 4:15-16"/>
  <object work="lxx" ref="Isa 8:23-9:1"/>
</claim>
```

<claim> indicates that a claim or assertion is being made.[8] <subject>, @verb, and <object> correspond to RDF's subject, predicate, and object. The arbitrary value of @verb is defined in the <head> by one or more IRIs. Each @work's value is defined by the corresponding source file, and can also be converted to IRI values. The value of @ref points to a specific part of one

[5] [Link to other article.]

[6] The TLG is problematic for identifying works, because some TLG numbers legitimately identify multiple works (e.g., any TLG number assigned to fragments), and some works are split across TLG numbers. For more concerns see Kalvesmaki 2014.

[7] Because this example uses the TLG's numbering system, the version of Isaiah being pointed to is the Greek Septuagint, and it may not lead to other versions of the Old Testament without explicit instructions to equate this with other versions of Isaiah.

[8] All claims must be attributed to a claimant, specified elsewhere in the TAN file. This departs from the RDF model, out of a concern that statements, assertions, and claims be attached to the person or entity making the claim.

or more source files that are part of the same work (as defined by the work IRI). The meaning and implications of @ref are calculated via the context of the relevant portions of the source TAN files.

Any TAN/TEI file or fragment may be algorithmically converted to a CTS URN. Unfortunately, the converse is not true.[9] Because the underlying TAN/TEI transcriptions are devoted to a single version of a single work in a single, standard reference scheme, the values in @ref are unique and are therefore just as reliable as any other method of pointing. Further, because every division in a TAN/TEI transcription must be defined as a particular kind of division, a computer can draw semantic inferences, e.g., that Mt and Isa both identify books, and that the numbers correspond to chapters and verses.

To refer to spans of text, the TAN syntax is more than are CTS URNs. For example, @ref may point to a discontinuous ranges of text, and it may also adopt numeration systems other than Arabic numerals :[10]

```
<claim verb="quotes">
  <subject work="src1" ref="4 b 5, 5c"/>
  <object work="src2" ref="12 b-d, f, 15 c"/>
</claim>
```

Ranges may be abbreviated economically, which results in a clean syntax that avoids repetition. That is, Mt 4:15-16 is sufficient. You don't need to write Mt 4:15-Mt 4:16. It is assumed that the first part of a range bestows upon any abbreviated second part the contextual hierarchy by which it can be resolved.[11]

There are many other features that could be added to this example (e.g., certainty, or adverbial modifiers), but explaining them would transfer us from the topic of pointers to that of RDF. We save that much-needed discussion for another day.

3 Version-specific Pointers

In dealing with Example B, pointers in the CTS syntax must be supplied with subreferences, and would be marked like this, with additions emphasized:[12]

- urn:cts:greekLit:tlg0031.tlg001:4.15-4.16
- urn:cts:greekLit:tlg0527.tlg048-**rahlfs**:8.23**@χώρα**-9.1**@ὑμᾶς**

The first insertion specifies through an arbitrary value some specific version. Nothing prevents one project from using **-rahlfs**, another, **-lxx**, and another, **-1935**, raising questions about the interoperability of a subreferenced URN. The last two insertions specify beginning

[9] Any CTS URN is a compound of items, each of which in the TAN format requires its own IRI, or resolution into one. For example, "4.15" point to two different types of divisions, but the CTS URN gives no indication of what kind of thing "4" and "15" are. TAN requires that all reference numbers be tethered to textual divisions that are defined with an IRI. For other comments on CTS URNs, see Kalvesmaki 2014.

[10] TAN permits the use of Arabic numerals, Roman numerals, alphabetic numerals, and combinations of Arabic and alphabetic numerals (e.g., 4a, D1). During validation all numerals are converted to an Arabic equivalent, and are evaluated on that basis (a combinatory numeral is converted to two Arabic numerals; the two preceding examples would become 4 and 1). Because of problems of ambiguity, Roman numerals may not be mixed with alphabetic numerals within the same file.

[11] One may legitimately object that in the phrase "1.3-4" it is unclear whether one means 1.3 and 1.4 or 1.3 through the entirety of 4. The TAN validation rules mimic how I believe we already read such references. We assume 1.4, unless that reference cannot be located in at least one version of the work. In that case we treat it as 4. If that also cannot be found then the references is treated as invalid.

[12] On CTS subreferences, see http://www.homermultitext.org/hmt-doc/cite/cts-subreferences.html (accessed 2017-10-18).

and ending points for the range of text. CTS allows character-specific references, so the second URN could be rendered:

- urn:cts:greekLit:tlg0527.tlg048-**rahlfs**:8.23**@[85]**-9.1**@[110]**

This character-based reference assumes space normalization, and precomposed Unicode characters. A CTS-based RDF rendering of Example B would look something like this:

urn:cts:greekLit:tlg0031.tlg001:4.15-4.16 http://purl.org/spar/fabio/Quotation
urn:cts:greekLit:tlg0527.tlg048-**rahlfs**:8.23**@χώρα**-9.1**@ὑμᾶς** .

The TAN approach to Example B builds upon that of Example A. We assume that we want to point to a portion of a text node that is within a leaf element (any element that contains text and no other elements). I have adopted the principle that TAN syntax should whenever possible mimic real-life practice. I have observed that in ordinary conversation when we want to point to a specific part of text, we prefer to talk about the nth (e.g., first, seventh, last) paragraph, sentence, line, word, or character. When we get down to the word level, we might identify the word by position (e.g., "the fifth to last word") or by name (e.g., "the last 'from'").

TAN adopts space normalization as a default, which means that text can be indented as freely as one wishes. So a text node's new lines are no guide to identifying what we normally mean by "line" or "paragraph." Therefore, those terms cannot be reliably used to identify a portion of a TAN text node. The meaning of "sentence" is contested, and difficult to define computationally: the most common sentence separator (at least in modern Western languages), the period, is used for too many other purposes.

So we are left with words and characters. TAN integrates both into its pointer syntax. But each of those concepts are ambiguous, and need some discussion before we look at real examples.

The term "word" is notoriously difficult to define. In different contexts, "New York" and "didn't," for example, can each be justifiably defined as one or two words. Furthermore, some scholars consider punctuation to be words in their own right (e.g., linguists' study of modern English), whereas others ignore them as being anachronistic or arbitrary (e.g., classicists' study of ancient Greek and Latin). TAN adopts the adjunct term "token": a string of characters that are defined by a regular expression meant to approximate what we usually mean by "word." And when I ask you to look at the fifth word, you normally ignore any intervening punctuation. Therefore, TAN's default definition of a token is [\w‍]+ — one or more consecutive word characters, the soft hyphen, the zero-width space, or the zero-width joiner. Although that is the default, it is not The Law. In any TAN file, an editor is welcome to write their own regular expression that defines what a token is.

"Character" also has some issues, even if we agree, as I think we should, that it should refer to a notional Unicode codepoint, whether expressed in UTF-8, UTF-16, UTF-32, or an HTML or XML entity (e.g.,
). The main problem is that Unicode is complicated, and has had to differentiate three levels of support for regular expressions.[13] Only level-one conformance is guaranteed in any XML system. And combining characters fall in level two. Suppose we have a string of three characters: ạb (i.e., a, a combining underdot, and a b). If I ask you via a regular expression to fetch me the second character, will you return the dot or the b? That decision, at present, depends upon the implementation you are using, and the

[13] http://www.unicode.org/reports/tr18/ (accessed 2017-10-17).

results are not predictable. Therefore, TAN defines a character as any non-modifying character along with any ensuing modifying characters (regular expression: \P{M}\p{M}*).

With that background, we can now look at how Example B could be rendered in TAN:

```
<claim verb="quotes">
  <subject work="nt-grc" ref="Mt 4:15-16"/>
  <object src="lxx">
    <tok ref="Isa 8:23" pos="15-last"/>
    <div-ref ref="Isa 9:1"/>
  </object>
</claim>
```

The principal change is that we are pointing to a range in Isaiah 8:23 that extends from the fifteenth word to the last. (During validation TAN functions silently convert the term "last" to a digit.) Note that @work has become @src, since we are now making a version-specific reference.

Suppose we have another situation where we wish to refer to a specific word or specific set of characters. Some arbitrary examples:

<tok ref="Isa 8:23" val="χώρα"/> (the first instance of the word)
<tok ref="Isa 8:23" val="καὶ" pos="2"/> (the second instance of the word)
<tok ref="Isa 8:23" pos="last" chars="1-2, last-2"/> (returns three characters from the last word, i.e., Ioí from Ιουδαίας)

Note that "last-2" means the second-to-last character—it is oftentimes so much easier to count from the end than from the beginning.

Because the values are resolved by means of regular expressions, @val may take a regular expression:

<tok ref="Gen 4:15" val="θε[όὸ]ς" pos="2"/>

This example points to the second instance of θεός regardless of whether the accent is grave or acute.[14]

Of course, if the value of the attributes point to something that does not exist, the TAN validation routines will mark the element as invalid, and via Schematron Quick Fixes offer contextual suggestions on how to fix the error. In fact, it is possible to ask for help by placing three question marks in an attribute then running a validation check. Either the error report itself or an SQF will provide editing help. So a version-specific pointer in TAN is not only readable, it is a kind of heuristic, at least when used during validation.

4 Pointing Problems

Such ambitious pointing has problems. But the problems I have found most difficult pertain to the concept of textual pointing in general. We might fuss about them in our digital projects, but they were there ever before there was a computer.

[14] TAN also permits an extension to regular expressions that makes it easier to identify classes of characters. In our example, one could write θε\k{.omicron}ς. I hope to publish more on this regular expression extension in the future.

One of the the general problems is that a reference system might not match across different versions of the same work. The Psalms, for example, have a significantly different numeration system in the Greek and Hebrew versions (Kalvesmaki 2014), which means that any reference you find to a Psalm, whether you read that reference on the screen or in print, could confuse you. A computer fares no better. What this means is that even within the same notional, version-agnostic reference system you are not guaranteed the interoperability we strove for when we invented those systems. In practice many of them fall short, to one degree or another. That is why I have used terms above like « try », « maximize », and « maximally likely », because I recognize that nothing can guarantee that texts produced by different people (or even the same person) will be in sync. TAN, however, creates a framework that makes such interoperability more likely, and provides diagnostics to help make our texts more interoperable.

TAN validation can be set to any of three phases—terse, normal, and verbose—to allow for variations in speed and depth.[15] In the verbose phase of a TAN alignment file, you will be told where there are references that are defective (references found in some but not all versions of the same work). Sometimes you can and should do something about defective references ; other times you shouldn't because a source is fragmentary or an editor has sequestered some passages. Verbose validation will give you the opportunity to review the differences. In that same TAN alignment file you can write statements that synchronize versions. For example, one source may have used « Matt » and another « Mt » for the same book. Or, in the case of Isaiah 8 :29, the versification of the Brenton and Rahlfs versions are out of sync. You can reconcile such differences through specially designated elements.

Another pointing problem is the case where a work has multiple, overlapping reference systems. For example, citations from the works of Aristotle rely upon both book-chapter references and Bekker page-column-line numbers. Citations from the Church Fathers oftentimes need both book-chapter references and volume-column-subcolumn references from Migne's Patrology.

TEI handles this problem by allowing users to supplement a transcription's primary hierarchy with <milestone>s, which usually mark page or column breaks. The <milestone> is very useful, but it lacks the requisite semantics for representing a textual hierarchy, as required under TAN rules.[16]

I have approached the problem differently. First, only one reference system may be applied to a TAN file. For each reference system desired, users should create a separate file with an identical transcription that is structured according to that reference system. These files should point to each other, which then makes reconciliation relatively straightforward. The verbose level of TAN validation allows one to check to make sure that the transcriptions are identical. If they are not, the specific differences are marked, and SQFs can be used to semi-automate the correction of the text.

Under this approach, a text can take as many flavors of hierarchies as you wish. And a cluster of differently segmented transcriptions can help TAN-based applications migrate texts from one reference system to another. In fact, I have been able to write an application that

[15] As of this writing (2017-10-19), the three-phase validation described here is still under development, but is planned to be released in the near future.

[16] The reasons are rather technical, but can be pointed to with a simple example of what a <milestone> should look like if it were to have the necessary semantics: <milestone unit="bk chap v" n="Mt 6 9"/>. This example is officially invalid because TEI disallows spaces in @unit. The TEI rules could, of course, be customized, but that would require us to move on to other technical problems. For example, <milestone>'s @n is a concatenation of several individual @n's. But 's @n allows spaces. So in the example <milestone unit="ch poem" n="Prelude One Treasure"/>, how would the hierarchy be restored? Are we talking about a chapter called Prelude One, or a poem called One Treasure?

takes as input a transcription pair (a single version of a work in two files, each with its own reference system) and a third file that contains another version of the same work, perhaps in a completely different language, but styled according to one of the reference systems. The output is the third file restructured in the alternative reference system, with a best guess as to where the divisions should fall.[17]

There are of course other problems about pointers that could be discussed. Some of them are treated in the official TAN guidelines and examples. Others not covered by the guidelines will be incorporated as they are encountered.

IV. Progress and Prospects

At present, TAN is an experimental format, under development. It has been used successfully in service to three different projects : the Guide to Evagrius Ponticus (GEP), the Chrysostomus Latinus in Iohannem Online project (CLIO), and a private collaborative effort to translate a fourth-century Greek work and its ancient Syriac translations (PRV).[18]

The GEP relies upon TAN files to generate and publish HTML transcriptions of select primary sources. It also takes a TAN file that contains RDF-like claims, along with a bibliography database (in Zotero RDF), and automatically generates an extensively documented master checklist of writings from the fourth century monk, creating links between the TAN files and the relevant bibliographic entries.

CLIO, which is focused on the three Latin translations of eighty-eight homilies by John Chrysostom on the Gospel of John, has developed a small library of master TAN-TEI files that are used to populate a website correlates the translations (and textual variants) with manuscript images. The TAN files include both straightforward transcriptions and indications of textual variants. It is hoped that in the future other projects treating the same work but in other languages (e.g., Greek, Armenian, Syriac, Arabic) will be able to create comparable TAN files that synchronize with the CLIO files, to demonstrate genuine cross-project interoperability.

For its work, PRV has depended upon a library of TAN files to prepare a book that is under contract with Oxford University Press. To reach that goal, PRV has developed a number of TAN-based XSLT tools that create various forms of output that have been useful not only for the published project but for conducting initial work. That output includes :

(1) Master HTML pages that collate every version (Greek, Syriac, the working English translations, etc.) not just with each other but with juxtaposed commentary and quotations from Aristotle and the Bible (in different versions). The number and sequence of the versions are easily configured, without touching the underlying TAN files.
(2) Master indexes of textual quotations from the Bible and Aristotle, sortable in the order of the quoted or quoting work, with contextual snippets.
(3) Master subject indexes.
(4) Comparative statistical profiles of each version of each work (e.g., word counts, hapax legomena, most frequent words).
(5) Reports that suggest how words in one particular language-version correspond to words in the others.
(6) Word-by-word interlinear editions.

[17] This application has yet to be published.

[18] GEP, director Joel Kalvesmaki, Dumbarton Oaks, http://evagriusponticus.net; CLIO, director Chris Nighman, Wilfrid Laurier University, https://www.chrisnighman.com/; PVT, director Robin Darling Young, Catholic University of America.

(7) Quotation detection.[19]

Every one of these applications would merit its own article-length discussion. As time and opportunity permits I will do so, but in the meantime, do explore the examples that have already been posted.[20]

Such progress is promising, but tentative. At the present, TAN needs people from other established projects in the digital humanities to use it and to actively contribute to the development of the schemas, examples, and documentation. By expanding the network, to see what rules work, what rules don't, and what can't be reduced to a rule, the TAN format will become more servicable, and provide insight into what kinds of rules can and cannot be reduced to an algorithm.

Trying it out incurs little risk, since TAN is not an exclusive format. A TAN file may sit happily side by side with TEI, HTML, plain text, or other formats. But it is primed to do things those other formats cannot do, namely, participate in a cross-project network, an ecosystem of peer-to-peer sharing of our sources and our annotations on those sources. Developing TAN has made me dream about someone to come along and create a Napster for digital texts.[21] TAN would be perfectly suited to a Text Napster. But whether or not such dream software materializes, I aim to continue developing TAN over the coming years, since I believe that even on its own it heralds a new Internet of primary sources.

References

Cayless, H. Rebooting TEI Pointers, *Journal of the Text Encoding Initiative* [Online], Issue 6 | December 2013, accessed 17 October 2017. URL: https://jtei.revues.org/907; doi:10.4000/jtei.907.

Kalvesmaki, J. Canonical References in Electronic Texts: Rationale and Best Practices, *Digital Humanities Quarterly*8.2 (2014), http://www.digitalhumanities.org/dhq/vol/8/2/000181/000181.html.

Kalvesmaki, J. "Three Ways to Enhance the Interoperability of Cross-References in TEI XML." Presented at Symposium on Cultural Heritage Markup, Washington, DC, August 10, 2015. In *Proceedings of the Symposium on Cultural Heritage Markup*. Balisage Series on Markup Technologies, vol. 16 (2015). doi:10.4242/BalisageVol16.Kalvesmaki01. http://www.balisage.net/Proceedings/vol16/html/Kalvesmaki01/BalisageVol16-Kalvesmaki01.html

Schmidt, D. Towards an Interoperable Digital Scholarly Edition, *Journal of the Text Encoding Initiative* [Online], Issue 7 | November 2014, Online since 12 November 2014, connection on 24 March 2015. URL:http://jtei.revues.org/979; doi:10.4000/jtei.979.

Schmidt, D. The Inadequacy of Embedded Markup for Cultural Heritage Texts, *Literary and Linguistic Computing* 25 (2010): 337-356. doi:10.1093/llc/fqq007.

[19] Because elsewhere in this issue we have a discussion of TRACER **[link to other article]**, the tool meant to help find textual reuse, I will mention that I have been able to leverage TAN files for textual reuse detection in two ways. In one method, TAN-formatted texts serve are piped into TRACER. Because TRACER output lacks the necessary metadata, creating results in the TAN format has proved difficult. In another approach, I wrote a XSLT stylesheet that mimicked what I believed TRACER was doing. This stylesheet takes as input any number of groups of TAN transcriptions; the output is a set of TAN annotations that show likely overlap between texts from any two groups. Texts processed through that stylesheet tended to take about 40% longer than the same texts processed by TRACER, but it resulted, at least for my test examples, in much richer results, which allowed me to identify not merely quotations but also thematic overlap.

[20] In addition to the links above, see the sample libraries and output posted at http://textalign.net/.

[21] Napster existed from June 1999 through July 2001 as a peer-to-peer music-sharing service. Users of the software shared access to locally stored music libraries (and equally important the metadata for that music). The service was shut down for lawsuits over copyright infringement. But for those of us whose primary work is in texts that are not subject, or should not be subject, to copyright law, the original Napster concept is ripe for revival.