

Supporting linguistic annotation using XML and stylesheets*

Jean Carletta, David McKelvie, and Amy Isard

University of Edinburgh

Large-scale linguistic annotation is currently employed for a wide range of purposes, including comparing communication under different conditions, testing psycholinguistic hypotheses, and training natural language engines. Current software support for linguistic annotation is poor, with much of it written for one-off tasks using special purpose data representations and data handling routines. This impedes research because software cannot be reused and the resulting annotations can be difficult to use in analyses or applications for which they were not originally intended. This paper argues for a particular vision of how support for linguistic annotation could be provided using XML as the data format and stylesheets as the processing mechanism.

1. Introduction

High-speed computing has enabled large-scale linguistic annotation in service of a wide range of research methods and applications. However, research progress is hampered by the lack of infrastructure for annotation technologies; much annotation is supported with special-purpose tools, making it difficult either to develop new coding systems or to place multiple annotations on the same source. We describe the range of reasons people have for annotating corpora, and the kinds of annotations that they perform. We then review the tools that have been used to support annotation for more than one particular set of codes. We describe how linguistic annotation can be supported using XML as the data format and stylesheets as the processing mechanism, and explain why we favour this approach.

2. What is Linguistic Annotation and What is it Used For?

By *linguistic annotation*, we mean any sort of annotation which one might like to add to linguistically derived data. This might mean adding free-format *notes* to specific points or spans in the data, or it might mean applying systematic *codes*, which can then be analysed statistically. Although there are several communities performing linguistic annotation, researchers in one are not always aware of researchers in another because they do not use data in the same way. Perhaps the largest scale use of linguistic annotation is in speech and language engineering, where annotations are typically used to build predictive models for natural language applications. Modellers can either be engineering-biased, looking for any easily automatable codes which help predict other phenomena for a particular corpus (for instance, many current approaches to “message understanding”, Chinchor, 1998), or theory-biased, where the features chosen are intended to have wider applicability, and the model, explanatory power (e.g., Shriberg et al., 1998). Where the modelling is exploratory rather than hypothesis-driven, purely statistical techniques such as machine learning, Markov modelling, or data mining may be employed to impose their own structure on the data.

* This work was funded at the Language Technology Group of the University of Edinburgh by grant GR/L29125 (NSCOPE) from the Engineering and Physical Sciences Research Council (UK), ROPA award R022250210 from the Economic & Social Research Council (UK), and projects LE4-8370 (MATE) and IST-2000-26095 (NITE) funded by the European Commission.

Correspondence concerning this article should be addressed to Jean Carletta, Human Communication Research Centre, Language Technology Group, University of Edinburgh, 2 Buccleuch Place, Edinburgh EH8 9LW, Scotland. Electronic mail may be sent to J.Carletta@edinburgh.ac.uk.

Linguists with a stronger theoretical bias may use very similar, cross-annotated data, but conduct their investigations inferentially, with research hypotheses determined ahead of time and formally tested (e.g., Bard et al., 2000). Psychologists also use linguistic annotations if they are studying language (e.g., Levelt, 1983, on disfluency) or working in areas where hypotheses can be tested by looking at language differences (e.g., Doherty-Sneddon et al., 1997, on the effects of video-mediation; Gottman, 1979, on dysfunctional marriages). Finally, researchers within traditions arising from the humanities often need to mark up linguistic data (Weitzman & Miles, 1994), but are more likely to restrict themselves to qualitative analyses, with the notes as an *aide memoire*, than researchers from other disciplines (but see Silverman, 1993). Like language engineers, they also annotate automatically, based on pattern-matching in the surface-form of the text with which they are working.

Just as the users of linguistic annotation are diverse, so are the kinds of annotations that are performed. Some annotation is of phenomena which one would properly describe as linguistic: syntax, for instance, or pragmatic information such as dialogue moves reflecting a speaker's intentions and the games that collect them by discourse goal. This annotation is not necessarily confined to that which can conveniently be attached to orthographic transcription, but may include, for instance, phonological or intonational information that requires access to speech waveforms. Other linguistic annotation is of allied phenomena that are not properly linguistic but are theoretically related to linguistic phenomena (e.g., kinesics). These may require access to video as well as to speech. Still other annotations are of interest for natural language and speech processing (for instance, coughing even when it is not intended to convey meaning, areas of high background noise, and speech recogniser output). Finally, most corpora require documentation about the data that they contain, which can most conveniently be distributed as part of the corpus itself in the form of headers, as the Text Encoding Initiative¹ (TEI) recommends (Sperberg-McQueen & Burnard, 1994). In addition to information about the corpus, such headers can usefully contain information about the annotations that have been performed upon it. Adequate software support must consider all of these types of annotation.

Each of these disciplines and types of annotation places somewhat different strains on support technologies. For instance, in the psychological tradition, researchers tend to restrict themselves to simple annotations tailored carefully to a particular set of hypotheses (Bakeman & Gottman, 1997). Data re-use is discouraged, except occasionally when data is re-analysed in order to test a new theory. Because the data is quite simple, once the hypothesis has been tested, there is nothing further to be done with it. On the other hand, empirical linguists prefer to annotate many phenomena on the same material and re-use it for several purposes. Visualization and purely exploratory modelling, although taking a similar approach to data collection, require much larger amounts of data than hypothesis-driven research. Researchers from the humanities need annotation that can be flexibly defined, often without much computing infrastructure, but may not need large-scale handling. Researchers who wish to annotate spoken waveforms or videos require much more complex capabilities than those who can work simply from orthographic transcription or written text.

Despite these differences, each tradition using linguistic annotation requires the same basic support. For speech, some sort of transcription usually comes first, followed by attachment of free-format notes and/or systematic codes. As well as

¹ See <http://etext.lib.virginia.edu/TEI.html>.

supporting transcription, annotation, and coding, software needs to support correction of everything that has previously been done to the speech data; coding often reveals errors in other codings applied to the same data as well as to the base transcription. It must also be possible to write routines that code data automatically. Finally, there must be tools which aid data analysis, whether they are data displays which help the user explore the codes present, decision-support tools to aid theory-building, or packages for graphing, statistics, and visualisation.

3. The requirement for data models that reflect natural structures

Software supporting linguistic annotation needs a data model that reflects the structure of the annotation and by which the data can be manipulated. The most primitive structure possible, which we think of as “unstructured” or “flat” coding, is one that simply associates tags with either orthographic spans of a written text or timed segments of a speech or video signal. A data model that only supports the addition and deletion of tags against such spans suffices for some purposes. However, the prototypical annotation structure, familiar from syntax, is the tree. Trees allow different tags to be associated with each other explicitly. In flat coding, it is possible for, say, an S to run from time t_0 to t_2 , an NP to run from t_0 to t_1 , and a VP to run from t_1 to t_2 , but that does not in itself mean that the NP and the VP together make up the S. The co-occurrence of tags could be a coincidence, or their relationship might be an empirical matter, not a theoretical one. For this reason, when working with tree structures, it is necessary for the data model to allow hierarchical relationships to be represented explicitly. Only by employing a data model that represents the structure of a linguistic annotation can its meaning be made clear.

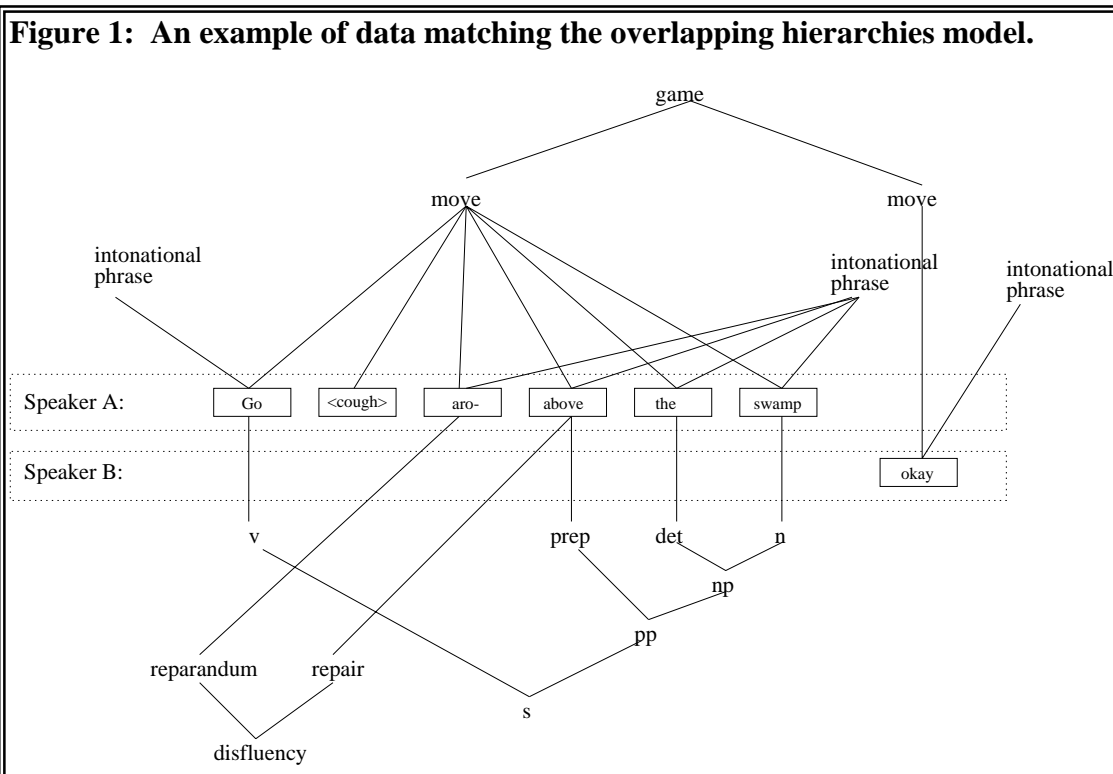
Although computers can work with many alternate representations of the same data, human data users need to be able to see and manipulate data in terms of the structure that they find most natural for it. Two examples will suffice to show that structure is important to human users. First, think of the data analyst wishing to know how prosody and dialogue structure are related. His research questions might include whether prosodic features differ at game-internal move boundaries from those at game boundaries, and whether the intonation of a reply move is affected by the type of the enclosing game. Since the analyst's questions are formed around the structure that he perceives in the data, the most natural way for him to pose these queries in the interface is by reference to that structure. This suggests that structured data models are important for end users, at least those with some theoretical interests. (Those who only wish to train on the statistical patterns in the data can do so without phrasing specific queries or visualizing the results.) Second, think of someone trying to add syntactic coding to a text corpus using a graphical user interface. Unless he is able to perceive syntax trees readily in the interface, he will have to carry this structural information in his head, making the coding process even more cumbersome and error-prone than usual. That is, annotation structure must be apparent in the data models employed in all human interfaces to the data if software is to be usable.

4. Which data model?

This human need leads to a problem for supporting linguistic annotation: if people are choosy about their structural representations, but every corpus is a one-off design, what kinds of data models should software employ? It is, of course, easiest to support simple structures. Efficient algorithms for working with flat lists and for tree traversal, for instance, are well understood; general graph traversal is not. Although it may seem fairly harmless to have tools that do not support the full range of structures

that humans might see in a data set, they can be subtly damaging to linguistic theory. Users will often maintain polite fictions about their theories for the sake of the support that the software provides (for instance, treating dialogue moves as if they segment utterances, even though this is incorrect in cases where people finish each other's utterances). Although it is important for tools to work efficiently and correctly, it is also important for them to support structures that are theoretically-motivated and not just those that are easy to implement.

The question, then, is whether there is any data model which suffices for most or all linguistic annotation and is less complex than supporting general graph operations. Rigidly hierarchical structures are common in linguistic theory, but even then, working with more than one kind of annotation on the same data requires one to



represent multiple, overlapping hierarchies that ultimately point to the same units aligned to a signal or text. Figure one gives an example of such a treatment, where the base unit is orthographic transcription and the annotations reflect dialogue structure, intonational phrasing, syntax, and disfluency without specifying the relationships among these types. The base unit can of course be anything that is sensible for the annotations being presented; research concentrating on aspects of the speech may wish to use phonetic transcription or regular short spans of the speech, for instance, whereas for some pragmatics research, the base unit might more simply be complete dialogue turns. The corresponding data model, then, supports tree operations, but allows for nodes to have multiple parents even though they must not have more than one set of children. This approach has been employed, for instance, in the MATE workbench (McKelvie et al., 2001).

Although many sets of annotations can be seen as overlapping and interlinked hierarchies, this data model does not suit all purposes. Not all linguistic theories lead to tree-structured codes. Lattices are used to represent the relationships among possible words heard in speech recogniser output. Directed graphs are useful for re-entrancy in grammatical structure and for linking orthographic transcriptions to

dictionaries. Even with agreement about what natural structures are required, knowing how best to handle them can be difficult. The “annotation graph” data model promoted by the LDC sees temporal ordering as the primary structure in the interests of efficiency (Bird & Liberman, 2001). The data handling in MATE takes the opposite approach, privileging traversal and manipulation of hierarchies. Meanwhile, Taylor et al. (2001) argue for the utility of redundant data representations that employ parallel, related data structures where the mapping between the structures is many-to-many but preserves the order of the elements in each. So representing the natural structure of a data set is important for human users, but it may be difficult to settle on a data model that both admits efficient implementation and suits a wide range of needs.

5. Other requirements

Although the need for a clear and appropriate data model is important, it is not the only requirement for good software support. Perhaps the toughest and most overlooked requirement is the need for flexible display and interface mechanisms. As is obvious to those who have tried to adapt existing software for their own coding purposes, a data display may work well for one data set or one task, but abysmally for another. One classic problem is that viewing sparse phenomena requires different browsing techniques from viewing common ones. Some phenomena need to be shown in context; for others, it is better to use the limited screen real estate to show more cases. For heavily cross-coded data, it can be difficult to differentiate all tag types typographically and at any rate, so much information at once may overload the human user. Just as different combinations of data and task require different displays, they also require different interface techniques for adding or changing the data. There are presumably some regularities to what makes a good interface technique for what task — for instance, menus and relatively short lists of enumerated values seem made for each other — and there have been experiments in the automatic construction of interfaces based on the type of coding task.² However, for the foreseeable future, the best chance for a usable interface is for a human to design it. That means what is required is not a one-size-fits-all tool, but support for this design and programming, so that users can create tools which fit the task easily, rather than relying on tools which do not fit or creating tailored tools from scratch.

In addition, software to support linguistic annotation must use well-understood, documented data formats, or at least be able to export information to such formats. When data formats are idiosyncratic, or worse, proprietary, this makes exchange of data difficult. Researchers must be able to communicate data to others who use different software packages, adopt better software when it comes along, and export data to already existing analytical packages such as Excel and SPSS. Incompatible data formats divide research communities and lead to reinvention of the wheel. Other things being equal and space permitting, it helps if data formats are inspectable, since then they are more amenable to manipulation beyond that envisioned by the original designers.

6. Current support for annotation

There are very many annotation support tools that have been built from scratch to support orthographic transcription for specific transcription conventions or data entry and display for specific coding schemes. Special-purpose tools are undoubtedly useful, but the lack of more generic support impedes research progress. People will

² Richard Tobin and Henry Thompson, personal communication.

always need to develop new transcription and coding systems because new theories require data before they can be tested. Even the same theory may require coding scheme modifications when applied to new data. For instance, an early workshop³ of the Discourse Resource Initiative found that many dialogue act schemes are only reliable on the corpora for which they were developed, and TOBI (Silverman et al., 1992), the most prominent prosodic coding scheme, is typically modified before it is applied to languages other than North American English (Jun, 2000; Mayo, Aylett, & Ladd, 1997). The use of tools which are hard-wired for particular annotations makes it difficult to compare different existing codings on the same data, and to develop new coding schemes, tempting people to use schemes which do not quite fit and making quality research impossible for those without easy access to their own computer programmers.

In addition to hard-wired tools, there are currently several software packages used to support various kinds of annotation where the tags themselves are not static. Four of these are used within the speech and language engineering communities.⁴ Xwaves⁵, among other functions, allows one to label spans of speech with text strings by identifying the correct span within a representation of the speech signal; this can be used for flat coding. Tags are stored in a separate file from the speech signal, which identifies the spans to which they apply using timestamps. GATE⁶ (Cunningham, Wilks, & Gaizauskas, 1996) has a similar functionality for texts and transcriptions instead of speech. Here, tag spans are identified by clicking and sweeping, and stored using character offsets from the start of the file. The tags may contain attribute-value pairs as well as simple types. Two further tools, N.b. (Flammia & Zue, 1995) and Alembic⁷ (Day et al., 1997), support the definition and use of tree-structured tag sets on texts and transcriptions and use SGML, a precursor of XML, as the storage format. It would be incorrect to suggest that none of the current tools allow any flexibility in setting up their displays and interfaces. They variously allow the user to change the display's colour map, turn on and off the display of individual tag types, decouple tag names from what appears on the interface menus, and define keyboard shortcuts. However, none of them allow the degree of flexibility that we have argued is essential for current research purposes.

In addition to these tools, there are a number of packages⁸ (e.g., ATLAS.ti, NUD*IST, and "The Ethnographer") arising out of the humanities for working with texts. These packages are interestingly different from what the speech and language community think of as annotation support tools, in that they support *ad hoc* notes and the development of systematic coding schemes as well as coding itself. Users tend to start with a wide set of fairly sparse codes, with structure emerging as the analysis progresses. Codes are either applied to presegmented text units or to arbitrary text spans, which may overlap each other. At any time, code sets can be structured and restructured, using query languages to explore the data coded so far and graphical user

³ Organized by Marilyn Walker, Lynette Hirschman, Johanna Moore and Aravind Joshi, and held at the University of Pennsylvania, March 1996. See <http://www.georgetown.edu/luperfoy/Discourse-Treebank/dri-kickoff.html>.

⁴ Any statement about the existence and functionality of software is bound to date quickly; the LDC keeps current information at <http://morph ldc.upenn.edu/annotation/>. At the time of writing, (Cappelli et al., 1998) is a useful resource for researchers looking for tools.

⁵ See <http://www.entropic.com/products/speechtechtoolkits.html>.

⁶ See <http://www.dcs.shef.ac.uk/research/groups/nlp/gate/>.

⁷ See <http://www.mitre.org/technology/alembic-workbench/>.

⁸ See <http://www.atlasti.de/index.html> for ATLAS.ti, or <http://www.scolari.co.uk/> for a range of tools supporting qualitative data analysis.

interfaces to describe the suspected relationships. Supported structures are usually hierarchical but can be as complicated as networks with user-defined types for links among the code types (c.f. ATLAS.ti). Although the research methodology supported by this software is very different from that in language and speech, the underlying technology can be quite similar, in that it relies on keeping track of annotated data and displaying it sensibly.

In addition to these tools, there has recently been work on supporting linguistic annotation not by providing tools themselves, but providing data models and interfaces that handle data storage, loading, and manipulation according to the models. Programmers can then employ these interfaces to make tool-writing easier. The annotation graph data model mentioned earlier is currently being supported in this way.⁹ The ATLAS project¹⁰ will be supporting a richer data model, but the model itself has yet to be specified at the time of this writing. To our knowledge, these projects leave programmers to develop their own display and interface routines.

7. The technology upon which our approach is based

Our approach to supporting linguistic annotation relies on augmenting existing XML technology. Before presenting our approach, we first introduce the base technology.

7.1 XML: A formal language for structured data representation

The Extensible Markup Language (XML) (Bray, Paoli, & Sperberg-McQueen, 1998), a development of the Standard Generalised Markup Language (SGML) (Goldfarb, 1990), is a meta-language for defining markup languages. A markup language is a way of annotating the structure of a class of text documents. XML is not same kind of language as the better known Hypertext Markup Language (HTML). HTML is a mark-up language that web browsers know how to interpret, whereas XML is a language in which languages like HTML can be formally defined. In XML, structural annotations are represented by means of tags, or *elements*, representing the information by virtue of their names and the attribute/value pairs associated with the element. For instance, the following example shows one way of marking up an XML element indicating the scope of a disfluency over an orthographic transcription of the speech. In the example, the XML markup is given in bold face. **Cough** and **disf** are the names of elements, where **cough** is intended to be interspersed with the orthographic transcription and **disf**, to operate over it. **Type** is the name of an attribute for the **disf** element, and **replacement**, the value for that attribute:

(1) Go **<cough/>** **<disf type="replacement">** aro- above **</disf>** the swamp.

Although not an absolute requirement when using the language, XML users can specify the document structure using either a *document-type definition* (DTD), as defined in the XML specification, or a *Schema* (Fallside, 2001). These are mechanisms for formally defining the allowable tags within a particular text and the allowable relationships among the different kinds of tags. Document structure definitions can be written to define any document type required. For instance, when marking up disfluencies along Levelt's (1989) theory, one might wish to specify that

⁹ See <http://www ldc.upenn.edu/AG/>.

¹⁰ See <http://www.nist.gov/speech/atlas/>. ATLAS is not related to ATLAS.ti.

the disfluency should contain a reparandum, followed by a moment of interruption, followed by an optional editing term, followed by a repair. This structure can be specified in, say, a DTD, by means of a context free grammar in which one specifies the allowable contents of any given type of element. Since each element has its own content model, DTDs specify a document's structure by hierarchical decomposition, starting with an element covering the entire document.

There are two data models that are commonly used when processing XML documents and supported by library implementations. The "Document Object Model", or DOM, represents an XML document as a tree structure.¹¹ The "Simple API for XML", or SAX, instead treats an XML document as a series of events such as start tags, content, and end tags.¹² Both are useful when writing applications that handle XML data.

7.2 *Stand-off annotation*

Although the main structure represented in a document structure definition is hierarchically decomposable, as is much of the structure proposed by individual linguistic theories, it is perfectly possible to represent non-hierarchical relationships among tags. For markup which has no normative rules about what tags a document will include or where the tags will appear, but where tags do not "cross" each other by requiring different decompositions of the same basic material, users can simply fail to specify the document structure. This is especially useful whilst developing a systematic coding scheme. More complex arrangements can be achieved through the use of *stand-off annotation*.

There are two basic approaches to stand-off annotation. The first, as utilized in the TEI (Sperberg-McQueen & Burnard, 1994), simply uses IDREFs within an annotation to specify where to find the associated base data. The second (Thompson & McKelvie, 1997) uses pointers, as defined by the upcoming XPointer specification (DeRose, Maler, & Daniel Jr., 2001), for the same purpose.¹³ Since the XPointer specification includes ranges, using pointers has the advantage that they allow one to associate an annotation with a range of base data, without having to identify every element within that range. When combined with XLink (DeRose, Maler, & Orchard, 2001), which provides a way of specifying links between documents, the second approach allows stand-off annotation to be placed in documents that are separate from the base data. This has the advantage on larger projects of allowing annotators to work on the same base data at the same time without creating data resolution conflicts. For instance, figure 2 gives the current XML file structure for the annotations pointing to one speaker's signal in the HCRC Map Task Corpus.

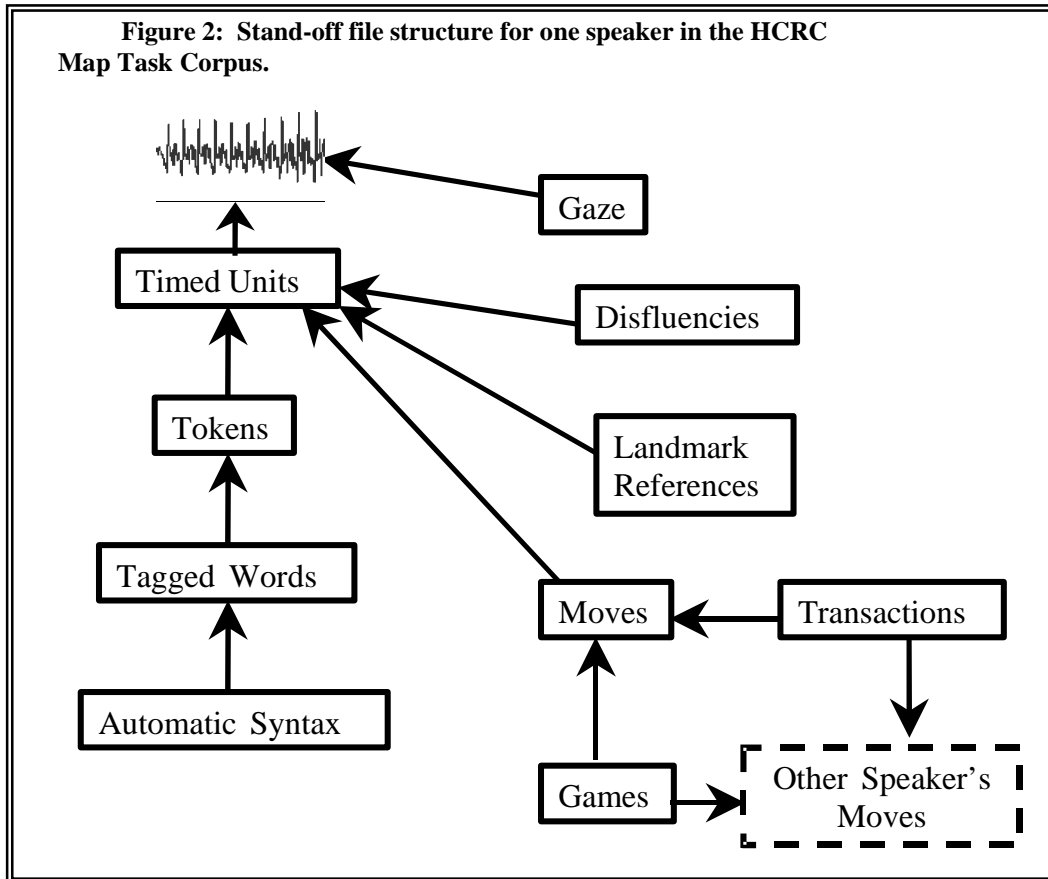
Stand-off annotation makes it possible to attach annotation not just to some form of orthographic transcription, but to some more general representation of the data, such as the timeline for the underlying speech. Note that in the data represented in figure 1, the orthographic transcription refers to time stamps from the speech.

¹¹ See <http://www.w3.org/DOM/>.

¹² See <http://www.saxproject.org/>.

¹³ XPOINTER is only a candidate recommendation at the moment, which means that it is possible for it to change or be revoked. However, it is already possible to implement stand-off annotation by this method because a number of software packages already support these and similar pointers. For instance, LT-XML (<http://www.ltg.ed.ac.uk/software/index.html>), which contains an integrated set of XML tools and a developers tool-kit, allows for this form of stand-off using preliminary versions of the XLINK and XPOINTER specifications. See <http://www.w3c.org/XML/Linking> for a list of implementations of the current specification.

Although in the example, all of the annotation is attached to the transcribed words or higher level phenomena, one can just as easily attach annotations directly to the speech timeline itself by using the same kind of representation adopted for the orthographic transcription. Of course, if one's data has video information synchronized with the speech, this representation will also serve for representing video annotations.



7.3 Stylesheets

XML is designed to be machine-readable. Ideally, it would rarely (if ever) be inspected directly. The Extensible Stylesheet Language (XSL) facilitates human-readable display of XML-encoded data. XSL has two parts: XSL Transformations (XSLT), and XSL Formatting (XSLF). XSLF consists of definitions specific to the typesetting of documents, and therefore is of little importance for linguistic annotation. XSLT, described in (Clark, 1999), is simply a language for defining transductions with XML data as input. Specifications expressed in XSLT are called *stylesheets*. A stylesheet can be used, for instance, to obtain HTML from any document conforming to the Text Encoding Initiative's DTD for novels. The transduction specified by this stylesheet might substitute an <H1> tag for the title, <H2> tags for chapter headings, and so on. Although this is a common use for XSLT, the transduction output can take any arbitrary form. Thus, other stylesheets for the same input might create a table of contents or a text-only list of words in the order in which they appear in the text. XSLT can also be used to transduce XML data into XML conforming to a different DTD, modifying the structure of the existing data.

Stylesheets in XSLT can be defined to work over any XML-encoded input. XSLT works by listing templates for the tags in a particular document that embody production rules for dealing with the data. Each template defines both requirements for the template to be considered a match and the output that matches should produce. For example, as might be useful when producing an HTML display of a book, the following template matches on chapter titles (represented in the TEI as the content of **head** tags contained within **div2** tags of type **chapter**) and inserts HTML tags around the title which mark them as second level headers:

```
<xsl:template match="div2[@type='chapter']/head">
  <H2>
    <xsl:apply-templates/>
  </H2>
</xsl:template>
```

The call to `apply-templates` simply styles the children of the **head** tag (in this case, the words in the title), ensuring that their HTML representation ends up within the H2 tag. Matches are expressed in the XPATH query language (Clark & DeRose, 1999), which can not only specify paths down the tree structure from the root of an XML document, but also many other kinds of matches, including ones which require access to other XML documents.

XSLT includes ways of defining global and local variables so that the exact output of a template can rely on the input, either from the template's match or from XML that has previously been processed. Templates in a stylesheet are applied in order of occurrence starting with the top element in the document. There is a mechanism for top-down left-to-right traversal of the input document hierarchy, and a default rule for unmatched elements.

8. Our proposal: how to support linguistic annotation

If all we needed were simple display for flat or rigidly hierarchical codes, XML and XSLT would already provide a basic engine for processing linguistic annotation. The structures supported by XPATH work well for these data models, and in our experience even people who would consider themselves novice programmers are able to write stylesheets that transform XML into the HTML representation of their choice. XML and XSLT are already supported by endless books, tutorials, and courses, and are widely used, with implementations of parsing, validity checking, and stylesheet processing in both freeware and commercial software both as stand-alone tools and as library routines for a range of programming languages.¹⁴ Therefore our proposal is to support the remaining linguistic annotation needs by extension of the stylesheet processing metaphor. This is exactly the approach taken in a number of recent software projects (Jacobson, Michailovsky, & Lowe, 2001; McKelvie et al., 2001; Milde & Gut, 2001). For this approach to work, further development is required in two areas: data modelling, and display and interface techniques.

We have already pointed out that it is essential for software to support data models and query languages that give a natural fit to the data, and described a number of models currently in use that do not match XML's inherent single document tree structure of the XPATH query language very well. The first step in supporting new data models is to define stand-off frameworks for them so that data that conforms to

¹⁴ See <http://www.oasis-open.org/cover/> for reliable and up-to-date information.

them can be represented in XML. This is a useful step anyway, since XML is increasingly the interlingua for data exchange. In addition to specifying the structure of the set of XML documents used to represent the data, metadata explaining how to interpret the stand-off must be defined telling, for instance, where to find the timestamps and what the XML links correspond to in the data model. The next step is to define appropriate query languages, necessary for working with the models in any case, and then to construct mappings from these query languages to XPATH expressions. This is sufficient for writing stylesheets that display data conforming to these data models; the user could write the stylesheet using the most natural query language, and then compile that into standard XSLT.

For data manipulation, the situation is more complicated, but not unmanageable. In this case, as well as the changes we have outlined, we would first need a mapping from operations in the data model to operations on the underlying stand-off XML. This would allow the data model API to be implemented. We would also need to be able to specify display and interface techniques as the output of the template matches. There are a number of ways to do this, but perhaps the best way is to define a generic, non-implementation dependent representation for how an interface should look and behave, which the stylesheet could construct. For instance, this representation (which could itself be expressed in XML) might allow the user to specify the use of display objects such as frames, textboxes, and menus of various sorts. Interface behaviours could be specified as being attached to the various forms of user input (mouse clicks, keyboard, and so on) and expressed in terms of the data model operations, with the interface updating itself during use. These display objects and interface procedures would then need to be implemented using one or more of the several platforms that support both XML processing and the construction of graphical user interfaces.

9. Conclusions

We have sketched what we see as the best method for supporting current software needs for linguistic annotation. Accepting our proposal depends, we think, on two things. First, it depends on assuming that, even if linguistic annotation is not rigidly hierarchical, there are hierarchical relationships prominent within it, so that tree operations play heavily in whatever the appropriate data model is. The less appropriate this assumption, the more onerous the task of mapping the data model and query language to XML and XPATH.¹⁵ We believe the assumption is appropriate to most linguistic annotation, but the closer one gets to analysing properties of the speech signal (as opposed to the language realized in the speech), the more it breaks down. Second, it depends on the implementation that we have sketched being workable, and, in particular, being fast enough to satisfy the human user in practice. This in turn depends at least in part on the data model required, since some admit more efficient handling than others. Although our sketch admittedly requires a reasonable amount of processing, it must be kept in mind that users can only work with a very limited amount of data at any one time, especially when compared to, say, the large corpora upon which batch operations are often performed. This question can best be settled by prototyping, which we have now set out to do.

¹⁵ For some query languages and operators, especially involving quantification, the mapping output would require some XSLT control structure as well. In the future, a better option in these cases might be mapping to XQuery (Boag et al., 2001), a database-oriented XML query language that is still under development.

10. References

- Bakeman, R., & Gottman, J. M. (1997). *Observing Interaction: An Introduction to Sequential Analysis* (second ed.). Cambridge: Cambridge University Press.
- Bard, E., Anderson, A., Sotillo, C., Aylett, M., Doherty-Sneddon, G., & Newlands, A. (2000). Controlling the intelligibility of referring expressions in dialogue. *Journal of Memory and Language*, 42, 1-22.
- Bird, S., & Liberman, M. (2001). A Formal Framework for Linguistic Annotation. *Speech Communication*, 33(1-2), 23-60.
- Boag, S., Chamberlin, D., Fernandez, M. F., Florescu, D., Robie, J., Siméon, J., & Stefanescu, M. (2001, 20 December). *XQuery 1.0: An XML Query Language* [W3C Working Draft]. W3C, <http://www.w3.org/TR/xquery/>.
- Bray, T., Paoli, J., Sperberg-McQueen, C. M., & Maler, E. (2000, 6 October). *Extensible Markup Language (XML) 1.0* [W3C Recommendation]. Retrieved from the World Wide Web: <http://www.w3.org/TR/REC-xml>.
- Cappelli, B., Dybkjær, L., Evert, S., Fitschen, A., Heid, U., Isard, A., Kipp, M., Klein, M., McKelvie, D., Mengel, A., Møller, M. B., & Reithinger, N. (1998). *MATE Deliverable D3.1: Specification of Coding Workbench*. EU Telematics Project LE4-8370 (Multilevel Annotation Tools Engineering), <http://mate.nis.sdu.dk/about/deliverables.html>
- Chinchor, N. A. (1998). Overview of MUC-7/MET-2. In N. A. Chinchor (Ed.), *Proceedings of the Seventh Message Understanding Conference (MUC-7)*. San Diego, CA: Science Applications International Corporation.
- Clark, J. (1999, 16 November). *XSL Transformations (XSLT) Version 1.0* [W3C Recommendation]. W3C, <http://www.w3.org/TR/xslt>.
- Clark, J., & DeRose, S. (1999, 16 November). *XML Path Language (XPath) Version 1.0* [W3C Recommendation]. W3C, <http://www.w3.org/TR/xpath>.
- Cunningham, H., Wilks, Y., & Gaizauskas, R. (1996). GATE - a General Architecture for Text Engineering, *Proceedings of COLING-96*. Copenhagen.
- Day, D., Aberdeen, J., Hirschman, L., Kozierok, R., Robinson, P., & Vilain, M. (1997). Mixed-Initiative Development of Language Processing Systems, *Fifth Conference on Applied Natural Language Processing*. Washington D.C., U. S. A.: Association for Computational Linguistics.
- DeRose, S., Maler, E., & Daniel Jr., R. (2001, 11 September). *XML Pointer Language (XPointer) Version 1.0* [W3C Candidate Recommendation]. W3C, <http://www.w3.org/TR/xptr/>.
- DeRose, S., Maler, E., & Orchard, D. (2001, 27 June). *XML Linking Language (XLink) Version 1.0* [W3C Recommendation]. W3C, <http://www.w3.org/TR/xlink/>.
- Doherty-Sneddon, G., Anderson, A. H., O'Malley, C., Langton, S., Garrod, S., & Bruce, V. (1997). Face-to-face and video-mediated communication: A comparison of dialogue structure and task performance. *Journal of Experimental Psychology-Applied*, 3(2), 105-125.
- Fallside, D. C. (2001, 2 May). *XML Schema Part 0: Primer* [W3C Recommendation]. W3C, <http://www.w3.org/TR/xmlschema-0/>.
- Flammia, G., & Zue, V. (1995). N.b.: A Graphical User Interface for Annotating Spoken Dialogue. In J. Moore & M. Walker & M. Hearst & L. Hirschman & A. Joshi (Eds.), *Working Notes from the AAAI Spring Symposium on Empirical Methods in Discourse Interpretation and Generation* (pp. 40-46). Palo Alto: AAAI.
- Goldfarb, C. F. (1990). *The SGML Handbook*: Clarendon Press.

- Gottman, J. M. (1979). *Marital interaction: Experimental investigations*. New York: Academic Press.
- Jacobson, M., Michailovsky, B., & Lowe, J. B. (2001). Linguistic documents synchronizing sound and text. *Speech Communication*, 33(1-2), 79-96.
- Jun, S.-A. (2000). K-ToBI (Korean ToBI) labelling conventions: Version 3. *Speech Sciences*, 7, 143-169.
- Levelt, W. J. M. (1983). Monitoring and self-repair in speech. *Cognition*, 14, 41-104.
- Levelt, W. J. M. (1989). *Speaking: From Intention to Articulation*. Boston, MA, USA: MIT Press.
- Mayo, C., Aylett, M., & Ladd, D. (1997). Prosodic Transcription of Glasgow English: An Evaluation Study of GlaToBI. In A. Botinis & G. Kouroupetroglou & G. Carayiannis (Eds.), *Proceedings of an ESCA Workshop: Intonation: Theory, Models and Applications* (pp. 231-234). Athens: ESCA and U. of Athens.
- McKelvie, D., Isard, A., Mengel, A., Møller, M. B., Grosse, M., & Klein, M. (2001). The MATE Workbench - an annotation tool for XML coded speech corpora. *Speech Communication*, 33(1-2), 97-112.
- Milde, J.-T., & Gut, U. (2001). The TASX-environment: an XML-based corpus database for time aligned language data. In S. Bird & P. Buneman & M. Liberman (Eds.), *Proceedings of the IRCS Workshop on Linguistic Databases* (pp. 174-180). Philadelphia: University of Pennsylvania.
- Shriberg, E., Bates, R., Stolcke, A., Taylor, P., Jurafsky, D., Ries, K., Coccaro, N., Martin, R., Meteer, M., & Van Ess-Dykema, C. (1998). Can Prosody Aid the Automatic Classification of Dialog Acts in Conversational Speech? *Language and Speech*, 41(3-4), 439-487.
- Silverman, D. (1993). *Interpreting qualitative data: methods for analysing talk, text and interaction*. London: Sage.
- Silverman, K., Beckman, M., Pitrelli, J., Ostendorf, M., Wightman, C., Price, P., Pierrehumbert, J., & Hirschberg, J. (1992). TOBI: A standard for labeling English prosody, *International Conference on Speech and Language Processing (ICSLP)*. Banff.
- Sperberg-McQueen, C. M., & Burnard, L. (1994). *TEI Guidelines for Electronic Text Encoding and Interchange (P3)*. Chicago and Oxford: ACH/ACL/ALLC Text Encoding Initiative.
- Taylor, P., Black, A., & Caley, R. (2001). Heterogeneous Relation Graphs as a Formalism for Representing Linguistic Information. *Speech Communication*, 33(1-2), 153-174.
- Thompson, H. S., & McKelvie, D. (1997). Hyperlink semantics for standoff markup of read-only documents, *Proceedings of SGML Europe*.
- Weitzman, E. A., & Miles, M. B. (1994). *Computer Programs for Qualitative Analysis*. Thousand Oaks CA: Sage.