A generic approach to software support for linguistic annotation using XML

Readings in Corpus Linguistics, ed. G. Sampson and D. McCarthy, London and NY: Continuum International, 2002. Originally circulated on the web in 2000.

A generic approach to software support for linguistic annotation using XML [*]

Jean Carletta, David McKelvie, Amy Isard

Human Communication Research Centre and Language Technology Group, University of Edinburgh

Andreas Mengel

Institut für Maschinelle Sprachverarbeitung, University of Stuttgart

Marion Klein

Deutsches Forschungszentrum für Künstliche Intelligenz, Saarbrücken

Morten Baun Møller

Natural Interactive Systems Laboratory, Odense University

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 handling routines. This impedes research because developing special purpose software is slow, and also makes it difficult to use existing annotations in analyses or applications for which they were not originally intended. XML, a text mark-up language which admits the possible annotations and allows reference to external files containing, for instance, speech and graphics, can be used as the basis of a representational format for linguistic annotation. XML is already a standard outside the linguistics community, and therefore is well-supported with basic processing software. It allows more formal and explicit representation of a wider range of possible annotation structures than formats currently in use. However, it can also be used for completely unstructured data or for data with an implicit structure which the annotators have yet to discover. Together with XSL, an emerging standard for XML transduction which makes it easier to display XML texts, adopting XML will enable faster tool development and more flexible data re-use.

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 which they perform. We then review the tools which have been used to support annotation for more than one particular set of codes. Annotation tools can be built

[*] This work was funded by EPSRC (UK) grant GR/L29125 (NSCOPE) and EC Telematics Project LE4-8370 (MATE). The writing of section 3 was made easier by the existence of the web page http://morph.ldc.upenn.edu/annotation/, for which we thank the Linguistic Data Consortium, although our set of features and implicit user communities are perhaps wider than theirs, and the consolidation of the material presented is our own.

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.

using a common annotation processing engine based on XML, a text mark-up language which can represent non-hierarchical structures via "stand-off" annotation. XML allows reference to, for instance, speech and pictures via links to external files. Because the language is already used widely for document processing, basic capabilities such as parsing and validity checking for it are already well-supported. We argue that adopting this common approach will enable faster tool development for the community as a whole and promote more complete research by making it easier to relate different annotations.

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 in the data, or it might be 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 aid prediction 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. 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., in press). 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 which are performed. Some linguistic annotation is of phenomena which one would properly describe as linguistic: syntax, for instance, or pragmatic information such as dialogue acts reflecting a speaker's intentions. 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 which requires access to speech waveforms. Other linguistic annotation is of allied phenomena which 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 processing (for instance, non-intentional coughing, areas of high background noise, and speech recognizer output). Finally, most corpora require documentation about the data which they contain, which can most conveniently be distributed as part of the corpus itself in the form of headers, as the Text Encoding Initiative[1] (TEI) recommends (Sperberg-McQueen & Burnard, 1994). In addition to information about the corpus, such headers can usefully contain information about the annotations which have been performed upon it. Software supporting linguistic annotation must handle all of these types of annotation properly if it is to be useful for all of the communities involved.

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 one or two annotations tailored carefully to a particular set of hypotheses. 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, corpus 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 which 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, orthographic transcription usually comes first, followed by attachment of free-format notes and/or systematic codes. As well as supporting transcription, annotation, and coding, software needs to support correction of everything which 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 be possible to write new routines which 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, descriptive statistics, inferential tests, or statistical techniques to find structure in the data.

3. Current support for annotation

There are very many annotation support tools which have been built 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 always need to develop new transcription and coding systems. New theories require different data before they can be tested. However, even the same theory may require coding scheme modifications when applied to new data. For instance, an early workshop[2] of the Discourse Resource Initiative found that dialogue act schemes are usually most 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 (Mayo, Aylett, & Ladd, 1997; Beckman & Jun, 1996). Although theory-motivated researchers strive for domain and language independent coding distinctions, the further removed from

[1] See http://etext.lib.virginia.edu/TEI.html.

[2] 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.

surface-level decisions, the less reliable a scheme will be (Bakeman & Gottman, 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 don't quite fit and making research impossible for those without easy access to their own computer programmers. There are currently several software packages used to support various kinds of annotation where the tags themselves are not prespecified.

3.1 ESPS/xwaves

Xwaves[3] is an environment for the analysis and manipulation of speech signals. As one of its many capabilities, it allows one to label spans of speech with text strings. This makes it a popular choice for supporting any data transcription and coding which requires time-alignment to the speech signal. Tags are stored in a separate file from the speech signal, identifying the span to which they apply. Xwaves itself imposes no structure on the set of tags for a speech file; each refers simply to a span using offsets from the start of the speech data file, and not to any other tags in the set.

3.2 GATE

GATE[4] (Cunningham, Wilks, & Gaizauskas, 1996) is an architecture which is meant to support natural language processing (NLP) applications by allowing users to implement automatic tagging procedures, hand-annotate the same information, and compare the results. Because of the NLP focus, it has no speech capabilities, although it is possible to use it on orthographic transcriptions of spoken language. GATE is based on the TIPSTER file architecture popularized by MUC.[5] Annotations are defined in a TIPSTER standard format and are identified by a basic type, with further attributes, if desirable. GATE stores annotations separately from the data being annotated, using character offsets from the beginning of the file to define a span, or set of spans, to which the annotation refers. As a result, the annotations themselves all point to the underlying data and are unstructured with respect to each other. There can be difficulties if the underlying data changes; this does not happen in most NLP applications, but can during manual annotation of speech, where closer listening during coding often reveals flaws in the basic transcription. GATE includes a tool for manual annotation in which the user selects an annotation type from a menu and sweeps out a span which the annotation covers; attributes are entered and corrections to the span or type made via a pop-op window. Existing annotations can be edited using a similar interface. The manual annotation tool can be configured to show where annotations occur on a display of the text, and to hide types not currently of interest. GATE also includes a tool for comparing hand and automatic annotations of the same annotation type. This tool does not show the base text, but displays the two annotation files in adjacent windows, with annotations which span the same text are aligned horizontally. This tool also performs some basic statistics comparing the two annotations.

[3] See http://www.entropic.com/products/speechtechtoolkits.html.
[4] See http://www.dcs.shef.ac.uk/research/groups/nlp/gate/.
[5] See http://www.itl.nist.gov/iaui/894.02/related_projects/tipster/.

Readings in Corpus Linguistics, ed. G. Sampson and D. McCarthy, London and NY: Continuum International, 2002. Originally circulated on the web in 2000.

3.3 N.b.

N.b. (Flammia & Zue, 1995) is a tool which supports the annotation of a tree-structured set of tags. It uses SGML, a predecessor of XML, as its file format. N.b. allows the user to specify a hierarchy of tags using its own tag description language, but the display of tags is fairly rigid. Indentation is used to convey hierarchical structure, and a user-definable list of colours corresponds to the tag types, with the user able to specify that some tags should not be shown. There is also one special-purpose display function for drawing parse trees. N.b. itself does not support analysis of the annotations, but it is possible to extract information from them using, for instance, the LT NSL library of SGML handling tools.[6] It does support very good connections to external programs to aid working with the data via a method of supporting external function calls, so that, for instance, one can make the display call other existing programs to view waveforms, listen to speech, and show related graphics.

3.4 The Alembic Workbench

The Alembic Workbench[7] (Day et al., 1997) is much like a cross between the annotation tools part of GATE and N.b.. Like GATE, it is primarily intended to support the comparison between automatic and manually-produced tags, but like N.b., it uses SGML as a file format and allows the user to define tag hierarchies, with the ability to configure the foreground and background colours used for different tags in the display. Alembic additionally allows some user interface configuration, by way of decoupling tag names from exactly what appears on the workbench menus and by allowing the user to specify their own keyboard shortcuts to the tags. Tagging comparisons are shown not by aligning different windows by text span, but by underlining spans with differences in a display of the source text. In addition to supporting annotation, Alembic supports extraction of text spans covered by named tags.

3.5 Software for qualitative data analysis

A number of software packages[8] (e.g., ATLAS.ti, NUD*IST, and "The Ethnographer") exist for performing qualitative data analysis, mostly within the humanities. They are usually used on written text. Although in speech terms, they cannot do more than note the timing of an annotation with respect to an audio file, some people use them for analysing speech, especially from transcription. 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 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

[6] See http://www.ltg.ed.ac.uk/software/index.html.

[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.

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.

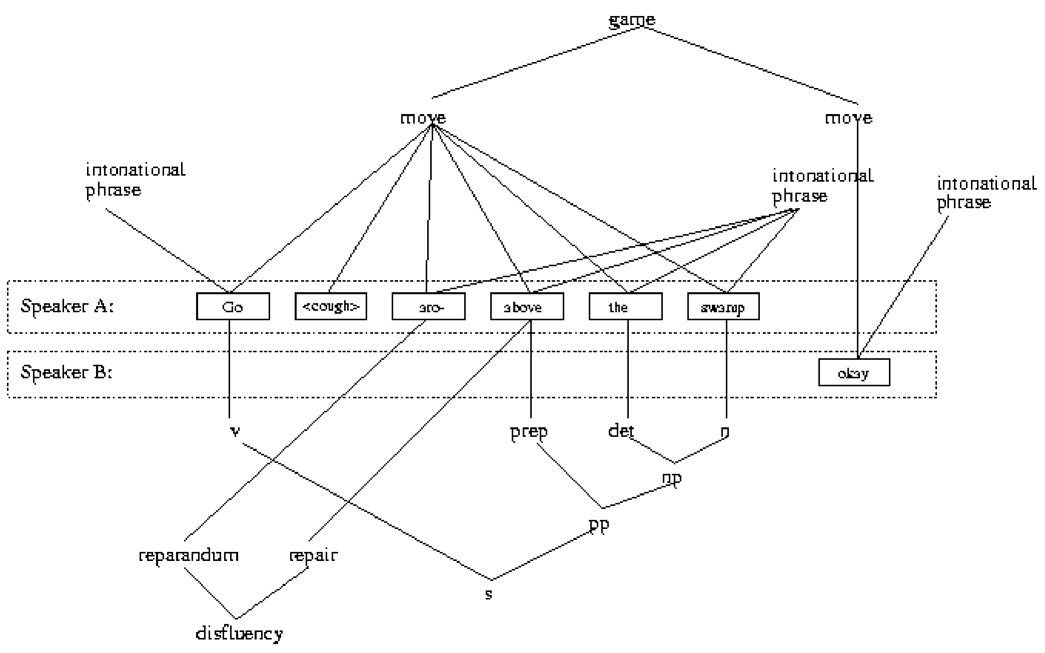
4. Requirement for integrated and structured (but not monolithic) support

Each of these coding architectures provides some degree of support for its adherents. However, the current support situation is limiting in several ways.

First, the use of idiosyncratic, and sometimes proprietary, data formats makes the exchange of data difficult. Apart from discouraging users from adopting better software when it comes along and from using analytical packages to which export was not predicted by the software developers, this also divides research communities. Software use spreads by word of mouth. Without the ability to look at data in the same way, communication across disciplines is limited, despite quite striking similarities in the annotations performed (e.g., that used by Eggins & Slade, 1997; for analysing casual conversations and the dialogue act codings used within computational linguistics, Carletta et al., 1997; Jurafsky, Shriberg, & Biasca, 1997).

Second, by failing to incorporate sufficient flexibility in how annotations are added to the data and displayed, current support still makes it difficult to establish new coding schemes. Although these architectures have come some way in decoupling exact tag names from display and coding methods, there is still work to be done on formal languages for relating tags to these methods, so that interfaces can be tailored to the user's needs.

Figure 1: An example of data matching the overlapping hierarchies model.



Third, architectures which support only tree-structured data may not be sufficiently general for current purposes. Some theories *are* rigidly hierarchical, but even then, working with more than one kind of annotation on the same data requires one to at least represent multiple, overlapping hierarchies cantilevered off the same units. 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 which 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.

However, not all linguistic theories lead to tree-structured codes. The more generic lattice is quite common; it is useful for representing, among other things, re-entrancy in grammatical structure and the relationships among possible words heard in speech recogniser output. Taylor et al. (1999) additionally argue for the relevance of parallel, related lists where the mapping between the lists is many-to-many but preserves the order of the elements in each. Although it may seem fairly harmless to have tools which do not support the full range of codings which one might wish to perform, they can be subtly damaging to linguistic theory. Users will often maintain polite fictions about their theories for the sake of the support which the software provides (for instance, treating dialogue moves as if they segment utterances, even though this is incorrect in the case of sentence completions). It is important for tools to support structures which are theoretically-motivated and not just those which are easy to implement. As theories develop and as the relationships among annotations becomes better understood, theories can change; for instance, in figure one, there may be a relationship between placement of disfluencies and syntactic structure which is yet to be discovered. Tools should therefore allow for the user to discover, formalize, and add new structures to the data, as suggested by ATLAS.ti.

Fourth, despite the inadequacy of tree-structures for linguistic annotation, software which treats tags as completely unstructured mark-up over spans of speech (c.f., Xwaves) or of transcription (c.f., GATE) is unhelpful. The key to providing good support is using a data model which humans can understand. Two examples will suffice to show that structure is important to human users. First, think of the data analyst wishing to know, for instance, 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 which 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, structure is also important earlier on, during the development of support tools. Software developers attempting to write usable interfaces to the data need to understand the structure which users will be expecting to see. Data sets where this structure is not reflected in the data manipulation and display methods imposes an extra burden on the human because it requires that information to be carried in the head.

Although for the sake of the humans working with the data, good support for linguistic annotation should include some specification of the data's structure, this structure need not necessarily be represented in the data's file format. One can build access methods to unstructured data which effectively impose the structure between the user and the data itself. Conversely, having structure present in the file format does not necessarily help the user unless the access methods also reflect that structure.

The important thing is for the *human* interface to the data to be structured as similarly to how they think of the data as possible. However, there is a good argument for making the structure at least discernible from the file format used to store the data. People like file formats which are inspectable, especially when they are writing software tools. Inspectable file formats at least give hints about how to work with the data efficiently and the chance of getting around the data model if it happens not to be as intuitive as the designer hoped. In addition, most access to the data will divide it along structural lines, and so computationally, sometimes having the structure present can improve efficiency. Inspectable file formats are not always possible — one of their drawbacks is that they take up more space — but a structured model of the data ought to reside somewhere, and there are benefits to reflecting it in the data format itself.

Note that, despite the limitations of current annotation support, the answer is not to build one integrated tool which supports all of the possible functions which every speech data user could possibly want. The needs of different kinds of research are so varied that such a tool would be unnecessarily cumbersome. In addition, some user functions, such as statistical analysis and display or annotation of individual properties based on the speech and not on some form of transcription, are so well supported by commercial software that a new, integrated tool would be very unlikely to supplant the current ones. Others, such as visualization and machine learning, may not yet be so well-supported, but are being developed in other disciplines. It is only the core functions which are currently poorly supported — adding linguistic annotations to a data source, displaying them, and manipulating, extracting or counting annotations which fit some given description. Implementors have shirked away from implementing general purpose tools which allow one to specify complex structures for the data because historically this has been difficult. Single structures are of course easier to represent and handle. In particular, algorithms for dealing with tree structures are well-developed. Alternatively, with unstructured tags there is no structure to represent, and at least then data can be maintained quite simply using standard storage techniques such as tries or hash tables. However, the World Wide Web Consortium (W3C) is now promoting a reasonable representational infrastructure for working with structured data in the form of XML.

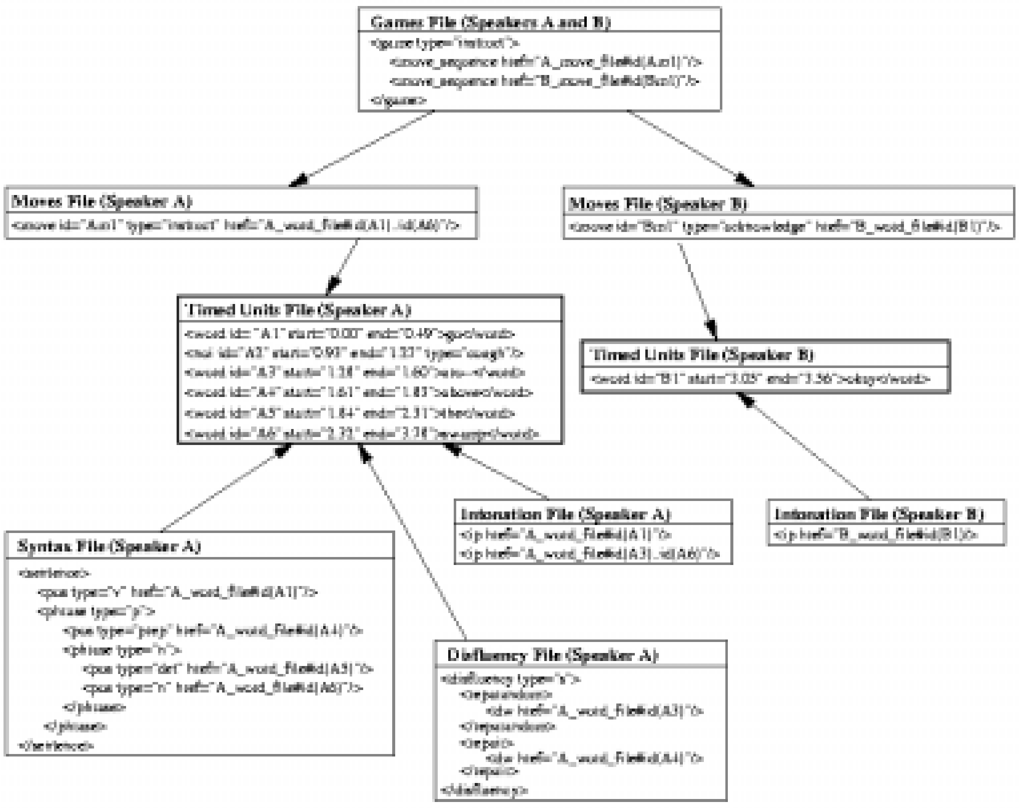
5. XML: A formal language for structured data representation

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 which 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 a *document-type definition* (DTD) which formally defines the tags which are allowable within a particular text and the allowable relationships among the different kinds of tags. One particularly popular DTD defines HTML, but DTDs 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 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 the 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.

Figure 2: Stand-off annotation for the data in figure one.


Games File (Speakers A and B)
Moves File (Speaker A)
Moves File (Speaker B)
Timed Units File (Speaker A)
Timed Units File (Speaker B)
Intonation File (Speaker A)
Intonation File (Speaker B)
Syntax File (Speaker A)
Disfluency File (Speaker A)

```
<game type="instruct">
   <move_sequence href="A_move_file#id(Am1)"/>
   <move_sequence href="B_move_file#id(Bm1)"/>
</game>
```

```
<move id="Am1" type="instruct" href="A_word_file#id(A1)..id(A6)"/>
```

```
<move id="Bm1" type="acknowledge" href="B_word_file#id(B1)"/>
```

```
<word id="A1" start="0.00" end="0.49">go</word>
<nsi id="A2" start="0.93" end="1.27" type="cough"/>
<word id="A3" start="1.28" end="1.60">aro-</word>
<word id="A4" start="1.61" end="1.83">above</word>
<word id="A5" start="1.84" end="2.31">the</word>
<word id="A6" start="2.32" end="3.28">swamp</word>
```

```
<word id="B1" start="3.05" end="3.56">okay</word>
```

```
<p href="A_word_file#id(A1)"/>
<p href="A_word_file#id(A3)..id(A6)"/>
```

```
<p href="B_word_file#id(B1)"/>
```

```
<sentence>
   <pos type="v" href="A_word_file#id(A1)"/>
   <phrase type="p">
      <pos type="prep" href="A_word_file#id(A4)"/>
      <phrase type="n">
         <pos type="det" href="A_word_file#id(A5)"/>
         <pos type="n" href="A_word_file#id(A6)"/>
      </phrase>
   </phrase>
</sentence>
```

```
<disfluency type="r">
   <reparandum>
      <dw href="A_word_file#id(A3)"/>
   </reparandum>
   <repair>
      <dw href="A_word_file#id(A4)"/>
   </repair>
</disfluency>
```

Although the main structure represented in a DTD 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 a DTD. This is especially useful whilst developing a systematic coding scheme. More complex arrangements can be achieved through the use of *stand-off annotation* (Thompson & McKelvie, 1997). In this approach, each data stream and each level of annotation is kept as separate XML coded files and aligned using links between elements which convey the intended structure. For instance, basic syntactic, intonational, disfluency, and dialogue structure annotation could be represented over the orthographic transcription in the example using stand-off annotation, as shown in figure two. In the figure, the syntax **href="B_word_file#id(B1)"** is a link to the **word** element with the id **"B1"** in the file **B_word_file**. This representation expresses the structure that dialogue games are made up of dialogue moves by having the **game** element point to **move** elements rather than pointing to **word** elements directly. Annotators whose do not wish to presuppose theoretical relationships such as the one embodied in this choice, are, of course, free to arrange the links among their elements differently. However, where such relationships are known to hold because of the theory underlying the annotation, it is undoubtedly easier to work with the data when the relationships are explicitly represented.

Although XML was developed for use in text formatting, 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 one, the orthographic transcription refers to time stamps from the speech. 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.

Although annotations can be stored using this representation, there is nothing in the XML shown which tells a support tool where to find external files in order to, for instance, play the speech or show the scope of an annotation against a display of a waveform. Speech and video could best be integrated into an XML data set if instead of relying on an external file with its own format they were also represented in XML. This would, however, take rather more diskspace than using one of the more traditional formats! The most likely solution is to provide information for processing the speech (such as how to put together the given start and end points into function calls for playing it) in a corpus header, making the information clear enough that applications can make use of it.

6. XSL: a formal language for representing formatting information

XML is designed to be machine-readable. Ideally, it would rarely (if ever) be inspected directly. The Extensible Stylesheet Language (XSL) has been developed to facilitate 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 which embody production rules for dealing with the data. Each template defines both requirements for the template to be considered a match and the output which matches should produce. 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 which 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.

7. Adapting XML and XSL for linguistic annotation

XSL is still under development. In addition, the main community of XML and XSL users will always have somewhat different needs than people using them to work with spoken language. As a result, there are a number of extensions to the current versions of XML and XSL which are required for linguistic work, some of which can be expected to occur in the main community and some of which must be provided within this one.

First, XSLT currently does not allow for any way of accessing the DTD structure of the input document. This means, for instance, that there is no way in the language to obtain a list of attributes for a particular tag. This makes writing style sheets cumbersome, and requires different style sheets even for very similar document types. Allowing access to the input DTD is a change with the W3C intends to make (Clark, 1999).

Second, XSLT is essentially a programming language, but a non-obvious one because the application order and precedence of the templates is rather complicated. This makes it somewhat difficult to write stylesheets, especially for those users without computational backgrounds. However, this difficulty presents itself as much outside the speech and language community as within it. Stylesheets are likely to develop into something easier to work with even without our attention to the problem, with software support to make them easier to write. Even as they stand, they are still an improvement over the alternative, which is having no formal language for decoupling data and display.

Third, because XSLT must match templates to parts of the document it is processing, it includes a formal query language for XML, which performs these matches. This query language currently focuses on detecting hierarchical structures within the document, in line with the most frequent applications for XML. Thus for strictly hierarchical linguistic annotations, the query language already expresses structure the human user can work with. The problem comes in dealing with linguistic annotations which cover several conflicting hierarchies or which are more accurately described using different basic structures. If people are to be able to extract meaningful information from the data or to write stylesheets for processing it, the query language must allow them to express the structures native to the data. Since speech annotations at their most basic at least usually have temporal ordering, it should also allow the expression of temporal relationships among the tags, especially where they are otherwise unstructured. Thus linguistic annotation requires the specification of a more flexible query language than is likely ever to be included in mainstream XML developments. Similarly, since the traversal mechanism for the XSLT transduction process is based on top-down, left-to-right processing, for linguistic applications which deviate greatly from the hierarchical norm, new traversal mechanisms may be more appropriate. Despite XML's privileging of tree structures, XML still provides a better foundation for the model of other data structures than do other architectures.

Finally, XSLT does not and is never likely to include the kind of support for interaction with a document via a user interface which is required for editing and coding. XSLT can be used to transduce the data into, for instance, objects in an object-oriented programming language which can then be manipulated in an interface and transformed back. However, for most XML applications there is no need for a standard language in which such an interface can be defined. It is nonetheless possible to define one. There are a limited number of well-supported techniques which one might use for interacting with displayed data, such as various kinds of menus, pop-up windows, and toolbars. Annotations themselves can be classified using a fairly small set of mathematical relationships expressing how they work over the data and over other annotations. This will certainly make it possible to specify coding and display interfaces abstractly. In fact, there are already experiments in generating usable coding interfaces for annotations automatically based on the coding scheme's DTD.[9]

8. Conclusions

Despite these further requirements, XML and XSL together provide a solid but flexible basis for working with linguistic data. XML is an inspectable file format, closely related to one which is already sometimes used for the annotation of transcribed speech, and allows for integration with external files and external processing routines. XML is sufficient for representing linguistic data as long as stand-off annotation is employed. XSL can be used for transduction to display outputs and for transforming the data to new structures as they are discovered. DTDs provide sufficient structure for querying the data and building transductions for data display and export when the tags form one or more hierarchies; more complex structures require some data modelling over the top of currently supported structures, but should still be easier to support upon this basis than upon a completely unstructured tag set. No current XML developments aid the formal specification of interfaces which allow the user to change XML data, but there is no work on such specifications for any other data encoding, either. None of the rival encodings to XML which have been used for linguistic data combine flexibility with formality of structural specification to the same degree.

In addition to the flexibility and structure which XML provides, XML has the advantage of already being in widespread use. It is increasingly being adopted as a data format for import and export in existing packages. There is every reason to believe that XSL will share XML's success. Just the fact that these languages are applicable standards for a wider community is reason enough for their adoption for linguistic annotation. Using them makes it unnecessary to develop a new language before the relationships between annotations in a particular corpus can be expressed. When developing languages from scratch, it is easy to get the essential basic properties wrong. In addition, because the community of XML users is much larger than the community of people who wish to perform linguistic annotations, XML is and will continue to be supported by a much better set of basic software tools than anyone in the linguistics community would be able to provide. For instance, although individual projects would be hard-pressed to provide such software for their own

[9] Richard Tobin and Henry Thompson, personal communication.

formal languages, a number of XML parsers and basic handling libraries already exist, along with graphical editors for DTDs, style sheets, and XML conformant to any given DTD.[10] XSL is itself an XML language, and developments within the W3C suggest that DTDs may themselves be restructured to be XML-conformant,[11] leaving open the possibility of writing editors for DTDs and stylesheets using XML technology itself. These tools cover the full range of platforms to which the varying user communities have access, and are expected to continue to do so. Although familiarising oneself with pre-existing formats and software takes time, using standard, widespread languages and tools is more efficient in the long run.

9. 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. (in press). Controlling the intelligibility of referring expressions in dialogue. Journal of Memory and Language.

Beckman, M. E., & Jun, S.-A. (1996). K-ToBI (KOREAN ToBI) Labeling Conventions (Working Papers in Phonetics (WPP) 966). Los Angeles, CA, USA: Ucla.

Bray, T., Paoli, J., & Sperberg-McQueen, C. M. (1998). Extensible Markup Language (XML) Version 1.0: http://www.w3.org/TR/REC-xml.

Carletta, J. C., Isard, A., Isard, S., Kowtko, J., Doherty-Sneddon, G., & Anderson, A. (1997). The reliability of a dialogue structure coding scheme. Computational Linguistics, 23(1), 13-31.

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). XSL Transformations (XSLT) Version 1.0: http://www.w3.org/TR/1999/07/WD-xslt-19990709.

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.

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.

Eggins, S., & Slade, D. (1997). Analysing Casual Conversation. London and Washington: Cassell.

[10] See http://www.oasis-open.org/cover/publicSW.html.
[11] See http://www.w3.org/, cf. schemas.

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.

Jurafsky, D., Shriberg, L., & Biasca, D. (1997). Switchboard SWBD-DAMSL Shallow-Discourse-Function Annotation Coders Manual, Draft 13 (Technical Report 97-02): University of Colorado Institute of Cognitive Science.

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 The University of Athens.

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. (1999). Heterogeneous Graphs as a Mechanism for Representing Linguistic Information (unpublished manuscript). Edinburgh: CSTR, University of Edinburgh.

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.