Chapter X

ENCODING SYNTACTIC ANNOTATION

Nancy Ide
Department of Computer Science, Vassar College, Poughkeepsie, New York 12604-0520, USA; ide@cs.vassar.edu

Laurent Romary
LORIA/CNRS, Campus Scientifique, B.P. 239, 54506 Vandoeuvre-les-Nancy, FRANCE; romary@loria.fr

Abstract. There is a need for a general framework for linguistic annotation that is flexible and extensible enough to accommodate different annotation types and different theoretical and practical approaches, while at the same time enabling their representation in a "pivot" format that can serve as the basis for comparative evaluation, merging, and the development of reusable editing and processing tools. To answer this need, we have developed a framework comprised of an abstract model for a variety of different annotation types (e.g., morpho-syntactic tagging, syntactic annotation, co-reference annotation, etc.), which can be instantiated in different ways depending on the annotator's approach and goals. The results have been incorporated into XCES (Ide, *et al.,* 2000a), the XML instantiation of the Corpus Encoding Standard (Ide, 1998a,b), which provides a ready-made, standard encoding format together with a data architecture designed specifically for linguistically annotated corpora.

Keywords. Corpus annotation standards, Extended Markup Language, Resource Description Framework

1 INTRODUCTION

Building a treebank requires several choices concerning annotation format. First, the annotator(s) must determine the *annotation scheme,* consisting of morpho-syntactic labels and syntactic constituent types together with general structural principles for the annotation, as dictated by the theory or model that informs the annotation. Second, the annotator must decide on an *encoding scheme,* that is, the physical representation of the annotation information in a physical document with tags, attributes, etc., as well as representation of both immediate and long-distance dependencies among constituents. Finally, the annotator must choose a *data architecture* for the primary text and its annotations, which dictates whether annotations are interspersed throughout the document containing the primary text or stored in one or more additional documents linked to the primary text. As the chapters in the previous section of this book demonstrate, these choices can differ considerably from project to project, even for the same language.

It is widely recognized that the proliferation of annotation schemes runs counter to the need to re-use language resources, and that standards for linguistic annotation are becoming increasingly mandatory. In particular, there is a need for a general framework for linguistic annotation that is flexible and extensible enough to accommodate different annotation types and different theoretical and practical approaches, while at the same time enabling their representation in a "pivot" format that can serve as the basis for comparative evaluation of parser output, such as PARSEVAL (Harrison, *et al.*, 1991), as well as the development of reusable editing and processing tools.

To answer this need, we have developed such a framework comprised of an abstract model for a variety of different annotation types (e.g., morpho-syntactic tagging, syntactic annotation, co-reference annotation, etc.), which can be instantiated in different ways depending on the annotator's approach and goals. We have implemented both the abstract model and various instantiations using XML schemas (Thompson, *et al.*, 2000), and the Resource Definition Framework (RDF) (Lassila and Swick, 2000) and RDF schemas (Brickley and Guha, 2000), which enable description and definition of abstract data models together with means to interpret, via the model, information encoded according to different conventions. The results have been incorporated into XCES (Ide, *et al.*, 2000a), the XML instantiation of the Corpus Encoding Standard (Ide, 1998a,b). XCES provides a ready-made, standard encoding format together with a data architecture designed specifically for linguistically annotated corpora. By exploiting the power of XML and RDF to implement an abstract model for instantiations of specific annotation schemes, XCES also provides a flexible and extensible mechanism for encoding a wide variety of linguistic annotations. The framework presented here for syntactic annotation is closely related to on-going work on the development of annotation schemes for terminology (Terminological Markup Framework[1], ISO project n.16642) and lexical data (Ide, *et al.*, 2000b).

2 XCES

The eXtensible Markup Language (XML) is the emerging standard for data representation and exchange on the World Wide Web (Bray, Paoli, & Sperberg-McQueen, 1998). Although at its most basic level XML is a document markup language directly derived from SGML (i.e., allowing tagged text (elements), element nesting, and element references), various features and extensions of XML make it a far more powerful tool for data representation and access. For example, the eXtensible Style Language (XSL) provides a powerful transformation language (XSLT) (Clark, 1999) that can be used to convert any XML document into another document (either another XML document or a document marked with HTML, etc.) by selecting, rearranging, and adding information to it, in order to serve any application that relies on part or all of its contents. Also, XML's provision for accessing part or all of multiple DTDs in a single document provides an elegant means to represent and manipulate multiple documents representing a text and its annotations.

XCES[2], the XML instantiation of the Corpus Encoding Standard (Ide, 1998a, b), is part of the EAGLES Guidelines developed by the Expert Advisory Group on Language Engineering Standards (EAGLES).[3] XCES is designed to be optimally suited for use in language engineering research and applications, in order to serve as a widely accepted set of encoding standards for corpus-based work in natural language processing applications. The standard specifies a minimal encoding level that corpora must achieve to be considered standardized in terms of descriptive representation (marking of structural and typographic information), provides a suite of DTDs and XML schemas for encoding basic document structure and linguistic annotation, and specifies a corresponding data architecture for linguistically annotated corpora.

[1] http://www.loria.fr/projects/TMF
[2] http://www.xml-ces.org
[3] http://www.ilc.pi.cnr.it/EAGLES/home.html

We are currently developing a comprehensive framework for linguistic annotation within the XCES which exploits the capabilities of the XML environment, with the ultimate goal of providing a fully-specified web-based format that enables maximal inter-operability among not only annotations of the same phenomena, but across annotation types. We aim to provide an environment in which annotations can be easily defined (and validated), rather than to dictate the use of specific annotation values, elements, etc. To this end, we are developing the following:

- a repository of existing annotation formats for a variety of linguistic features and, where necessary, a XML schemas to instantiate them together with XSLT scripts to transduce between different formats.
- XML schemas to instantiate a hierarchically specified structural model for annotations, beginning at the most abstract level and then defining derived types for general classes of annotation (e.g., speech, discourse, morpho-syntax, etc.). Precise annotation values will be specified in schemas at the lowest level of the hierarchy that can be used "off the shelf" by corpus annotators or modified to suit specific needs. Because types and sub-types are specified in an increasingly precise hierarchy, it is relatively trivial to back up one or more levels of abstraction and define new sub-types, and variant types can be easily created from existing ones by defining new derived or extended types.
- Because XML-encoded annotated corpora are increasingly used for interchange between individual processing and analytic tools, for commonly used tools we are developing XSLT scripts for mapping, and extraction of annotated data, import/export of (partially) annotated material, and integration of results of external tools into existing annotated data in XML.

3 SYNTACTIC ANNOTATION : CURRENT PRACTICE

At the highest level of abstraction, syntactic annotation schemes must represent the following kinds of information:

(1) *Category information*: labeling of components based on syntactic category (e.g., noun phrase, prepositional phrase), syntactic role (subject, object), etc.;

(2) *Dependency information*: relations among components, including constituency relations, grammatical role relations, etc.

For example, the annotation in Figure 1, drawn from the Penn Treebank II[4] (hereafter, PTB), uses LISP-like list structures to specify constituency relations and provide syntactic category labels for constituents. Some grammatical roles (subject, object, etc.) are implicit in the structure of the encoding: for instance, the nesting of the NP "the front room" implies that the NP is the object of the prepositional phrase, whereas the position of the NP "him" following and at the same level as the VP node implies that this NP is the grammatical object. Additional processing (or human intervention) is required to render these relations explicit. Note that the PTB encoding provides some explicit information about grammatical role, in that "subject" is explicitly labeled (although its relation to the verb remains implicit in the structure), but most relations (e.g., "object") are left implicit. Relations among non-contiguous elements demand a special numbering mechanism to enable cross-reference, as in the specification of the NP-SBJ of the embedded sentence by reference to the earlier NP-SBJ-1 node in the PTB example.

```
((S   (NP-SBJ-1 Jones)
      (VP followed)
      (NP him)
      (PP-DIR       into
             (NP the front room))
      ,
      (S-ADV (NP-SBJ *-1)
             (VP   closing
                   (NP the door)
                   (PP   behind
                         (NP him)))))
      .))
```

Figure 1 : Penn Treebank in-line annotation for the sentence "Jones followed him into the front room, closing the door behind him."

[4] http://www.cis.upenn.edu/treebank

Although they differ in the labels and in some cases the function of various nodes in the tree, most of the annotation schemes described in this volume provide a similar constituency-based representation of relations among syntactic components. In contrast, dependency schemes (e.g., Sleator and Temperley, 1993; Tapanainen and Jarvinen, 1997; Carroll, *et al.*, this volume) do not provide a constituency analysis[5] but rather specify grammatical relations among elements explicitly; for example, the sentence "Paul intends to leave IBM" could be represented as shown in Figure 2,where the predicate is the relation type, the first argument is the head, the second the dependent, and additional arguments may provide category-specific information (e.g., *introducer* for prepositional phrases, etc.). Although dependency schemes do not rely on hierarchical nesting etc. to indicate relations, an independently specified relation hierarchy may be defined which enables construction of a syntax tree from the dependency annotation.

```
subj(intend,Paul,_)
xcomp(intend,leave,to)
subj(leave,Paul)
dobj(leave,IBM,_)
```

Figure 2: Dependency annotation according to Carroll, Minnen, and Briscoe

4 A MODEL FOR SYNTACTIC ANNOTATION

The goal in the XCES is to provide a framework for annotation that is theory and tagset independent. We accomplish this by treating the description of any specific syntactic annotation scheme as a process involving several knowledge sources that interact at various levels. The process allows one to specify, on the one hand, the informational properties of the scheme (i.e., its capacity to represent a given piece of information), and, on the other, the way the scheme can be instantiated (e.g., as an XML document). Figure 3 shows the overall architecture of the XCES framework for syntactic annotation.

[5] So-called "hybrid systems" (e.g., Basili, *et al.*, 199; Grefenstette, 1999) combine constituency analysis and functional dependencies, usually producing a shallow constituent parse that brackets major phrase types and identifying the dependencies between heads of constituents.

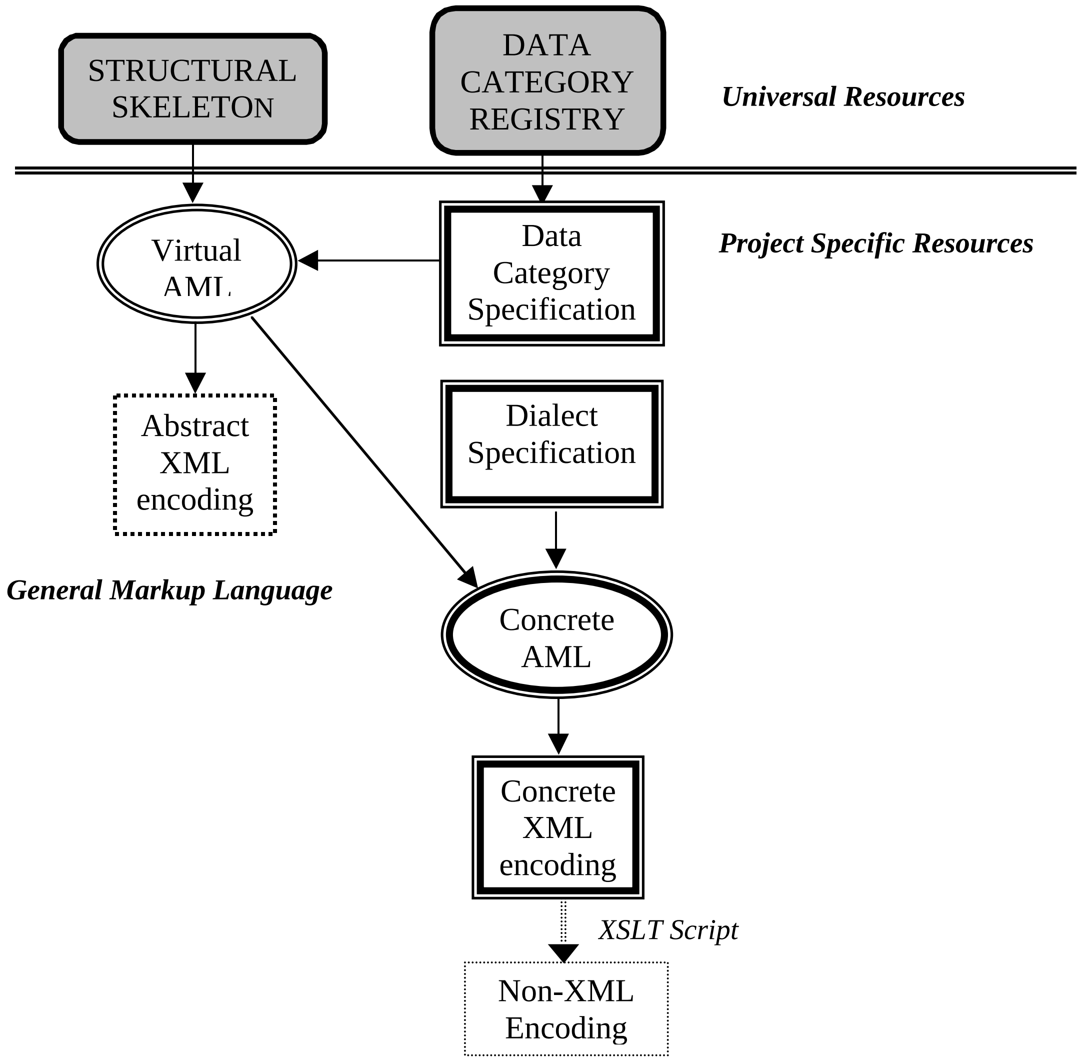

Figure 3 : Overall architecture of the XCES annotation framework

Two knowledge sources are used define the abstract model:

Data Category Registry: Within the framework of the XCES we are establishing an inventory of data categories for syntactic annotation, initially based on the EAGLES Recommendations for Syntactic Annotation of Corpora (Leech *et al.*, 1996). Data categories are defined using RDF descriptions that formalize the properties associated with each. The categories are organized in a hierarchy, from general to specific. For example, a general *dependent* relation may be defined, which may have one of the possible values *argument* or *modifier*; *argument* in turn may have the possible values *subject*, *object*, or *complement*; etc.[6] Note that RDF descriptions function much like class definitions in an object-oriented programming language: they provide, effectively, templates that describe how objects may be instantiated, but do not constitute the objects themselves. Thus, in a document containing an actual annotation, several objects with the type *argument* may be instantiated, each with a different value. The RDF schema ensures that each instantiation of *argument* is recognized as a sub-class of *dependent* and inherits the appropriate properties.

Structural Skeleton: a domain-dependent abstract structural framework for syntactic annotations, capable of fully capturing all the information in a specific annotation scheme. The structural skeleton for syntactic annotations is described below in section 4.1.

Two other knowledge sources are used to define a project-specific format for the annotation scheme, in terms of its expressive power and its instantiation in XML:

Data Category Specification (DCS): describes the set of data categories that can be used within a given annotation scheme, again using RDF schema. The DCS defines constraints on each category, including restrictions on the values they can take (e.g., "text with markup"; a "picklist" for grammatical gender, or any of the data types defined for XML), restrictions on where a particular data category can appear (level in the structural hierarchy—see section 4.1). The DCS may include a subset of categories from the DCR together with application-specific categories additionally defined in the DCS. The DCS also indicates a level of granularity based on the DCR hierarchy.

[6] Cf. the hierarchy in Figure 1.1, Caroll, Minnen, and Briscoe, this volume.

Dialect specification: defines, using XML schemas, XSLT scripts, and XSL style sheets, the project-specific XML format for syntactic annotations. The specifications may include

- *Data category instantiation styles* : Data categories may be realized in a project-specific scheme in any of a variety of formats. For example, if there exists a data category *NounPhrase*, this may be realized as an <NounPhrase> element (possibly containing additional elements), a typed element (e.g. <cat type=NounPhrase>), tag content (e.g., <cat>NounPhrase</cat>), etc.
- *Data category vocabulary styles* : Project-specific formats can utilize names different from those in the Data Category Registry; for instance, a DCR specification for *NounPhrase* can be expressed as "NP" or "SN" ("syntagme nominal", in French) in the project-specific format, if desired.
- *Expansion structures*: A project-specific format may alter the structure of the annotation as expressed using the structural skeleton. For example, it may be desirable for processing or other reasons to create two sub-nodes under a given <struct> node, one to group features and one to group relations (see section 4.1).

The combination of the structural skeleton and the DCS defines a **virtual annotation markup language (AML)**. Any information structure that corresponds to a virtual AML has a canonical expression as an XML document; therefore, the inter-operability of different AMLs is dependent only on their compatibility at the virtual level. As such, virtual AML is the hub of the annotation framework: it defines a *lingua franca* for syntactic annotations that can be used to compare and merge annotations, as well as enable design of generic tools for visualization, editing, extraction, etc.

The combination of a virtual AML with the Dialect Specification provides the
information necessary to automatically generate a **concrete AML** representation of the annotation scheme, which conforms to the project-specific specification provide in the Dialect specification. XSLT filters translate between the representations of the annotation in

concrete and virtual AML, as well as between non-XML formats (such as the LISP-like PTB notation) and concrete AML.[7]

4.1 THE STRUCTURAL SKELETON

For syntactic annotation, we can identify a general, underlying model that informs current practice: specification of constituency relations (with some set of application-specific names and properties) among syntactic or grammatical components (also with a set of application-specific names and properties), whether this is modeled with a tree structure or the relations are given explicitly.

Because of the common practice for syntactic annotation utilizing trees, together with the natural tree-structure of markup in XML documents, we provide a structural skeleton for syntactic markup that follows this approach. The skeleton consists of the following tags:

- *<struct>* represents a node (level) in the tree. <struct> elements may be recursively nested at any level to reflect the structure of the corresponding syntax tree.
- *<feat>* (feature) is used to provide information attached to the node in the tree represented by the enclosing <struct> element. A *type* attribute on the <feat> element identifies the data category of the feature. The tag may contain a string that provides an appropriate value for the data category (e.g., for *type=CAT* the value might be “NP”) or may point via a *target* attribute to an object in another document that provides the value. Note that this allows the possibility for including not only simple values but also complex data items as annotations. It also enables generating a single instantiation of an annotation value in a separate document that can be referenced as needed within the annotation document.

The <struct> element has the following possible attributes:

- *type* : specifies the node label (e.g., “S”, “NP”, etc.)
- *xlink* : points to the data to which the annotation applies. In the XCES, we recommend the use of *stand-off annotation*—i.e., annotation that is maintained in

[7] Strictly speaking, an application-specific format could be translated directly into the virtual AML, eliminating the need for the intermediary concrete AML format. However, specially for existing formats, it is typically more straightforward to perform the two-step process.

a document separate from the primary (annotated) data. The *xlink* attribute uses the XML Path Language (XPath) (Clark & DeRose, 1999) to specify the location of the relevant data in the primary document.

- *ref* : refers to a node defined elsewhere, used instead of *xlink.*
- *rel* : specifies a type of relation (e.g., “subj”)
- *head* : specifies the node corresponding to the head of the relation
- *dependent* : specifies the node corresponding to the dependent of the relation
- *introducer* : specifies the node corresponding to an introducing word or phrase
- *initial* : gives a thematic or semantic role of a component, e.g., “subj” for the object of a *by*-phrase in a passive sentence.

The hierarchy of <struct> elements corresponds to the nodes in a phrase-structure analysis; each <struct> element is typed accordingly. The grammar underlying the annotation therefore specifies constraints on embedding that can be instantiated in an XML schema , which can then be used to prevent or detect tree structures that do not conform to the grammar. Conversely, the grammar rules implicit in annotated treebanks, which are typically not annotated according to a formal grammar, can be easily extracted from the abstract structural encoding.

Figure 4 shows the annotation from the PTB (Figure 1) rendered in the abstract XML format. Note that in this example, relations are encoded only when they appear explicitly in the original annotation, and therefore heads of relations default to “unknown”. An XSLT script could be used to create a second XML document that includes the relations implicit in the embedding (e.g., the first embedded `<struct>` with category NP has relation “subject”, the first VP is the head, etc.). A strict dependency annotation encoded in the abstract format uses a flat hierarchy and specifies all relations explicitly with the *rel* attribute, as shown in Figure 5.[8]

[8] For the sake of readability, this encoding assumes that the sentence “Paul intends to leave IBM” is marked up as
`<s1><w1>Paul</w1><w2>intends</w2><w3>to</w3><w4>leave</w4><w5>IBM</w5></s1>`.

```
<struct id="s0" type="S">
    <struct id="s1" type="NP"
             xlink:href="xptr(substring(/p/s[1]/text(),1,5))"
             rel ="SBJ"/>
    <struct id="s2" type="VP"
             xlink:href="xptr(substring(/p/s[1]/text(),7,8))"/>
    <struct id="s3" type="NP"
             xlink:href="xptr(substring(/p/s[1]/text(),16,3))"/>
    <struct id="s4" type="PP"
             xlink:href="xptr(substring(/p/s[1]/text(),20,4))"
             rel="DIR">
     <struct id="s5" type="NP"
              xlink:href="xptr(substring(/p/s[1]/text(),25,14))"/>
    </struct>
    <struct id="s6" type="S" rel="ADV">
        <struct id="s7" ref="s1" type="NP" rel="SBJ"/>
        <struct id="s8" type="VP"
                xlink:href="xptr(substring(/p/s[1]/text(),41,7))">
            <struct id="s9" type="NP"
                     xlink:href="xptr(substring(/p/s[1]/text(),49,8))"/>
            <struct id="s10" type="PP" rel="DIR"
                     xlink:href="xptr(substring(/p/s[1]/text(),57,6))">
                <struct id="s11" type="NP"
                         xlink:href="xptr(substring(/p/s[1]/text(),64,3))"/>
            </struct>
        </struct>
    </struct>
</struct>
```

Figure 4 : The PTB example encoded according to the structural skeleton

```
   <struct rel="subj"  head="w2" dependent="w1"/>
   <struct rel="xcomp" head="w2" dependent="w4"  introducer="w3"/>
   <struct rel="subj"  head="w4" dependent="w1"/>
   <struct rel="dobj"  head="w4" dependent="w5"/>
```

Figure 5 : Abstract XML encoding for the dependency annotation in Figure 2.

5 USING THE XCES SCHEME

Despite its seeming complexity, the framework outlined in the previous section is intended to reduce overhead for annotators and users. Part of the work of the XCES is to provide XML support (e.g., development of XSLT scripts, XML schemas, etc.) for use by the research community, thus eliminating the need for XML expertise at each development site.

Also, with XML at its core, the XCES framework builds on emerging standards for data interchange for which there is widespread support; common scripts and tools can be developed, reducing the “reinventing of the wheel” that is commonplace in the area of corpus annotation at present. Because XML-encoded annotated corpora are increasingly used for interchange between individual processing and analytic tools, for commonly used tools we are developing XSLT scripts for mapping, and extraction of annotated data, import/export of (partially) annotated material, and integration of results of external tools into existing annotated data in XML. Tools for editing annotations in the abstract format, which automatically generate virtual AML from Data Category and Dialect Specifications, are already under development in the context of work on the Terminological Markup Language, and a tool for automatically generating RDF specifications for user-specified data categories has already been developed in the SALT project.[9] Several freely distributed interpreters for XSLT have also been developed (e.g., xt[10], Xalan[11]). As XML use becomes more widespread, more and more reusable tools and resources (including web-based applications) are becoming available. In practice, then, annotators and users of annotated corpora will rarely see XML and RDF instantiations of annotated data; rather, they will access the data via interfaces that automatically generate, interpret, and display the data in easy-to-read formats.

The abstract model that captures the fundamental properties of syntactic annotation schemes provides a conceptual tool for assessing the coherence and consistency of existing schemes and those being developed. The abstract model enforces clear distinctions between implicit and explicit information (e.g., functional relations implied by structural relations in constituent analyses), and phrasal and functional relations. It is alarmingly common for annotation schemes to represent these different kinds of information in the same way, rendering their distinction computationally intractable (even if they are perfectly understandable by the informed human reader). Hand-developed annotation schemes used in treebanks are often described informally in guidebooks for annotators, leaving considerable

[9] http://www.loria.fr/projets/SALT
[10] Clark, J., 1999. XT Version 1991105. http://www.jclark.com/xml/xt.html
[11] http://www.apache.org

room for variation; for example, Charniak (1996) notes that the PTB implicitly contains more than 10,000 context-free rules, most of which are used only once. Comparison and transduction of schemes becomes virtually impossible under such circumstances. While requiring that annotators make relations explicit and consider the mapping to the XCES abstract format increases overhead, we feel that the exercise will help avoid such problems and can only lead to greater coherence, consistency, and inter-operability among annotation schemes.

The most important contribution to inter-operability of annotation schemes is the Data Category Registry. By mapping site-specific categories onto definitions in the Registry, equivalences (and non-equivalences) are made explicit. Again, the provision of a "standard" set of categories, together with the requirement that scheme-specific categories are mapped to them where possible, will contribute to greater consistency and commonality among annotation schemes.

6 CONCLUSION

The framework presented here for syntactic annotation is intended to allow for variation in annotation schemes while at the same time enabling comparison and evaluation, merging of different annotations, and development of common tools for creating and using annotated data. We have developed an abstract model for annotations that is capable of representing the necessary information while providing a common encoding format that can be used as a pivot for combining and comparing annotations, as well as an underlying format that can be manipulated and accessed with common tools. The details presented here provide a look "under the hood" in order to show the flexibility and representational power of the abstract scheme; however, the intention is that annotators and users of syntactic annotation schemes can continue to use their own or other formats with which they are comfortable, and translation into and out of the abstract format will be automatic.

The XCES framework for linguistic annotation is built around some relatively straightforward ideas: separation of information conveyed by means of structure and information conveyed directly by specification of content categories; development of an abstract format that puts a layer of abstraction between site-specific annotation schemes and standard specifications; and creation of a Data Category Registry to provide a reference set of annotation categories. The emergence of XML and related standards, together with RDF, provides the enabling technology. We are, therefore, at a point where the creation and use of annotated data and concerns about the way it is represented can be treated separately—that is, researchers can focus on the question of *what* to encode, independent of the question of *how* to encode it. The end result should be greater coherence, consistency, and ease of use and access for linguistic annotated data.

References

Basili, R., Pazienza, M. T., Zanzotto, F.M. (1999). Lexicalizing a Shallow Parser. *Proceedings of TALN'99.* Corgese, Corsica, 1999.

Biron, P., Malhotra, A. (2000). XML Schema Part 2: Datatypes. W3C Candidate Recommendation, 24 October 2000. http://www.w3.org/TR/xmlschema-2/.

Bray, T., Paoli, J., Sperberg-McQueen, C.M. (eds.) (1998). Extensible Markup Language (XML) Version 1.0. W3C Recommendation. http://www.w3.org:TR/1998/REC-xml-19980210.

Brickley, D. and Guha, R.V. (2000). Resource Description Framework (RDF) Schema Specification 1.0. W3C Candidate Recommendation, 27 March 2000. http://www.w3.org/TR/rdf-schema/.

Carroll, J., Minnen, G., Briscoe, T. (2001). Parser Evaluation Using a Grammatical Relation Annotation Scheme. In Abeillé, Anne (ed.) *Treebanks: Building and Using Syntactically Annotated Corpora,* Kluwer Academic Publishers, this volume.

Clark, J. (ed.) (1999). XSL Transformations (XSLT). Version 1.0. W3C Recommendation. http://www.w3.org/TR/xslt.

Clark, J. and DeRose, S. (1999). XML Path Language (XPath). Version 1.0. W3C Recommendation. http://www.w3.org/TR/xpath.

Grefenstette, G. (1999). Shallow Parsing Techniques Applied to Medical Terminology Discovery and Normalization. *Proceedings of IMIA WG6, Triennial Conference on Natural Language and Medical Concept Representation*. Pheonix, AZ.

Harrison, P., Abney, S., Black, E., Flickinger, D., Gdaniec, C., Grishman, R., Hindle, D., Ingria, B., Marcus, M., Santorini, B., Strzalkowski, T. (1991). Evaluating syntax performance of parser/grammars of English. In *Proceedings of the Workshop on Evaluating Natural Language Processing Systems.* 29th Meeting of the Association for Computational Linguistics, Berkeley, CA, 71-77.

Ide, N. (1998a). Encoding Linguistic Corpora. In *Proceedings of the Sixth Workshop on Very Large Corpora*, 9-17.

Ide, N. (1998b). Corpus Encoding Standard: SGML Guidelines for Encoding Linguistic Corpora. In *Proceedings of the First International Language Resources and Evaluation Conference,* 463-70.

Ide, N., Bonhomme, P., Romary, L. (2000). XCES: An XML-based Standard for Linguistic Corpora.. *Proceedings of the Second Language Resources and Evaluation Conference (LREC),* Athens, Greece, 825-30.

Ide, N., Kilgarriff, A., Romary, L. (2000). A Formal Model of Dictionary Structure and Content. In *Proceedings of EURALEX'00*, Stuttgart, 113-126..

Lassila, O. and Swick, R. (1999) Resource Description framework (RDF) Model and Syntax Specification. W3C Recommendation, 22 February 1999. http://www.w3.org/TR/REC-rdf-syntax.

Leech, G, Barnett, R., Kahrel, P. (1996). EAGLES *Recommendations for the Syntactic Annotation of Corpora*. EAG-TCWG-SASG/1.8, 11th March 1996. http://www.ilc.pi.cnr.it/EAGLES/segsasg1/segsasg1.html.

Sleator, D., Temperley, D., (1993). Parsing English with a Link Grammar. *Proceedings of the 3rd International Confernece on Parsing Technologies, IWPT'93.*

Tapanainen, P., Jarvinen, T. (1997). A Non-projective Dependency Parser. *Proceedings of ANLP'97,*Washington D,C.

Thompson, H., Beech, D., Maloney, M. Mendelsohn, N. (2000). XML Schema Part 1: Structures. W3C Candidate Recommendation, 24 October 2000. http://www.w3.org/TR/xmlschema-1/.