A Common Framework for Syntactic Annotation

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

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

Abstract

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. To answer this need, we have developed a representation 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. In this paper we provide an overview of our representation framework and demonstrate its applicability to syntactic annotation. We show how the framework can contribute to comparative evaluation and merging of parser output and diverse syntactic annotation schemes.

1 Introduction

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 a representation 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), 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), part of the EAGLES Guidelines developed by the Expert Advisory Group on Language Engineering Standards (EAGLES)[1]. The XCES provides a ready-made, standard encoding format together with a data architecture designed specifically for linguistically annotated corpora.

In this paper we provide an overview of our representation framework and demonstrate its applicability to syntactic annotation. The framework has been applied to the representation of terminology (Terminological Markup Framework[2], ISO project n.16642) and computational lexicons (Ide, *et al.,* 2000b), thus demonstrating its general applicability for a variety of linguistic annotation types. We also show how the framework can contribute to

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

comparison and merging of diverse syntactic annotation schemes.

2 Current Practice

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

- *Category information*: labeling of components based on syntactic category (e.g., noun phrase, prepositional phrase), syntactic role (subject, object), etc.;
- *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[3] (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.

Although they differ in the labels and in some cases the function of various nodes in the tree, most annotation schemes provide a similar constituency-based representation of relations among syntactic components (see Abeille, forthcoming, for a comprehensive survey of syntactic annotation schemes). In contrast, dependency schemes (e.g., Sleator and Temperley, 1993; Tapanainen and Jarvinen, 1997; Carroll, *et al.,* forthcoming) do not provide a constituency analysis[4] 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.).

```
((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. PTB annotation of Jones followed him into the front room, closing the door behind him.

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

Figure 2. Dependency annotation according to Carroll, Minnen, and Briscoe (forthcoming)

3 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.

---

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

[4] 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.

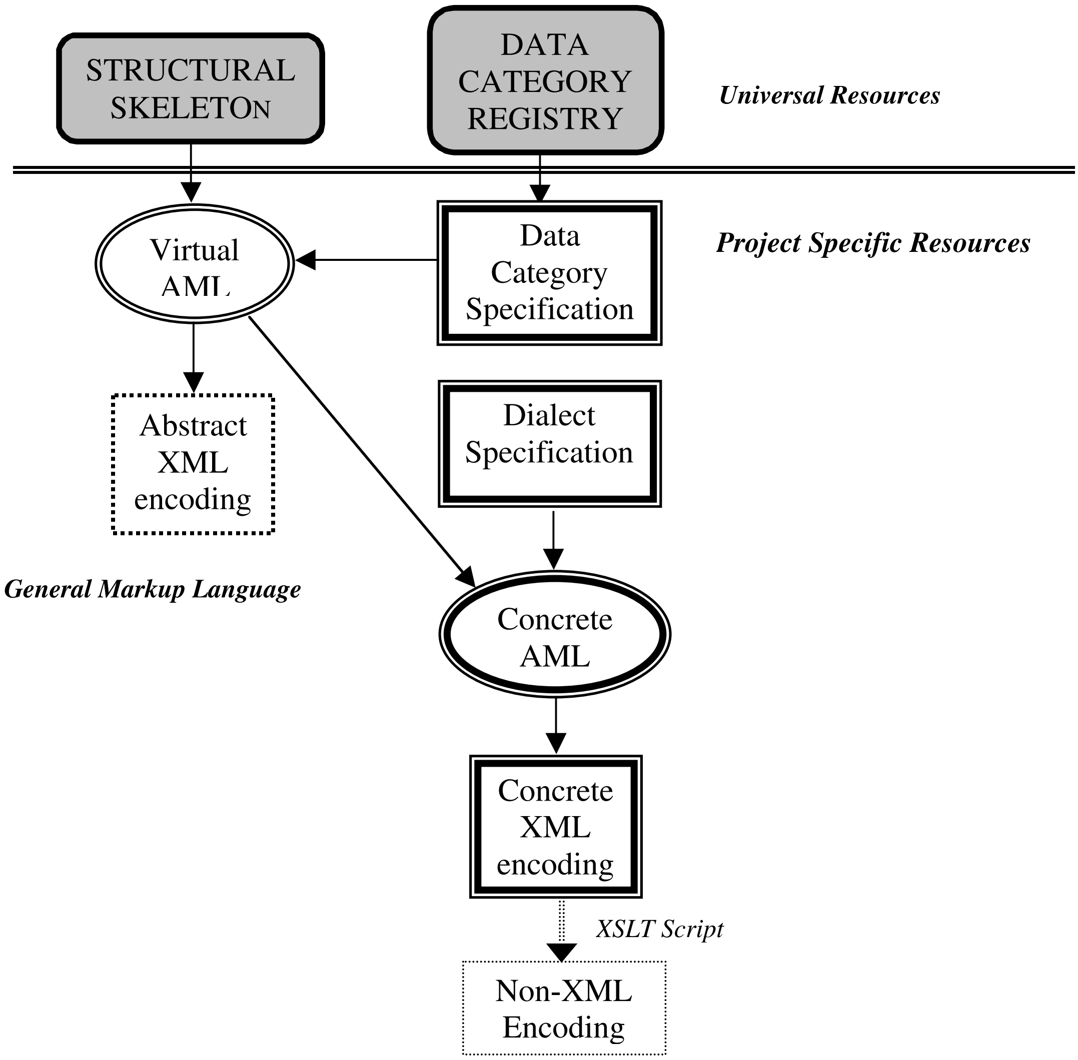

Figure 3. Overall architecture of the XCES annotation framework

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.[5] 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

[5] Cf. the hierarchy in Figure 1.1, Caroll, Minnen, and Briscoe (forthcoming).

annotations, capable of fully capturing all the information in a specific annotation scheme. The structural skeleton for syntactic annotations is described below in section 12.1.

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). 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.

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 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.

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 format provided 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.[6]

2.1 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 use of trees in syntactic annotation, together with the natural tree-structure of markup in XML documents, we provide a structural skeleton for syntactic markup following this model. The most important element in the skeleton is the *<struct>* element, which represents a node (level) in the syntax tree. <struct> elements may be recursively nested at any level to reflect the structure of the corresponding tree. The <struct> element has the following attributes:

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

- *type* : specifies the node label (e.g., S , NP , etc.) or points to an object in another document that provides the value. This allows specifying 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.
- *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 a document separate from the primary (annotated) data.[7] 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.

The skeleton also includes a <feat> (feature) element, which can be used to provide additional information (e.g., gender, number) that is attached to the node in the tree represented by the enclosing <struct> element. Like <struct>, this element can be recursively nested or can point to a description in another document, thereby providing means to associate information at any level of detail or complexity to the annotated structure.

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 (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]

4 Using the XCES Scheme

The Virtual AML provides a pivot format that enables comparison of annotations in different formats including not only different constituency-based annotations, but also constituency-based and dependency annotations. For example, the PTB annotation corresponding to the dependency annotation in Figure 2 is shown in Figure 6. Figure 7 gives the corresponding encoding in the XCES abstract scheme. It is relatively trivial with an XSLT script to extract the information in the dependency annotation (Figure 5) from the PTB encoding (Figure 7) to produce a nearly identical dependency encoding. The script would use rules to make relations that are implicit in the structure of the PTB encoding explicit (for example, the xcomp relation that is implicit in the embedding of the S phrase).

The ability to generate a common representation for different annotations overcomes several obstacles that have hindered evaluation exercises in the past. For instance, the evaluation technique used in the PARSEVAL exercise is applicable to phrase structure analyses only, and cannot be applied to dependency-style analyses or lexical parsing frameworks such as finite-state constraint parsers. As the example above shows, this

[7] The stand-off scheme also provides means to represent ambiguities, since there can be multiple links between data and alternative annotations.

[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>.

problem can be addressed using the XCES framework.

It has also been noted that that the PARSEVAL bracket-precision measure penalizes parsers that return more structure than exists in the relatively flat treebank structures, even if they are correct (Srinivas, *et al.,* 1995). XSLT scripts can extract the appropriate information for comparison purposes while retaining links to additional parts of the annotation in the original document, thus eliminating the need to dumb down parser output in order to participate in the evaluation exercise. Similarly, information lost in the transduction from phrase structure to a dependency-based analysis (as in the example above), which, as Atwell (1996) points out, may eliminate grammatical information potentially required for later processing, can also be retained.

```
((S   (NP-SBJ-1 Paul)
      (VP intends)
      (S     (NP-SBJ *-1)
             (VP  to
                 (VP  leave
                     (NP IBM))))
.))
```

Figure 6. PTB annotation of "Paul intends to leave IBM.

```
<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.

```
<struct id="s0" type="S >
        <struct id="s1" type="NP  target="w1
                rel="SBJ" head="s2"/>
        <struct id="s2" type="VP  target="w2"/>
        <struct id="s3" type="S >
                <struct id="s4" ref="s1"
                        rel="SBJ" head="s6"/>
                <struct id="s5" type="VP  target="w3">
                        <struct id="s6" type="VP  target="w4">
                                <struct id= s7  type="NP  target="w5"/>
                        </struct>
                </struct>
        </struct>
</struct>
```

Figure 4 : PTB encoding of "Paul intends to leave IBM."

5 Discussion

Despite its seeming complexity, the XCES framework is designed 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. Because XML-encoded annotated corpora are increasingly used for interchange between processing and analytic 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]). In practice, 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 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 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

[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

are mapped to them where possible, will contribute to greater consistency and commonality among annotation schemes.

6 Conclusion

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 such as 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 annotated data.

References

Anne Abeill (ed.), forthcoming. *Treebanks: Building and Using Syntactically Annotated Corpora,* Kluwer Academic Publishers.

Eric Atwell, 1996. Comparative evaluation of grammatical annotation models. In R. Sutcliffe, H. Koch, A. McElligott (eds.), *Industrial Parsing of Software Manuals,* 25-46. Rodopi.

Paul Biron and Ashok Malhotra, 2000. XML Schema Part 2: Datatypes. W3C Candidate Recommendation. http://www.w3.org/TR/xmlschema-2/.

Tim Bray, Jean Paoli and C. Michael Sperberg-McQueen (eds.), 1998. Extensible Markup Language (XML).

Dan Brickley and R.V. Guha, 2000. Resource Description Framework (RDF) Schema Specification 1.0. http://www.w3.org/TR/rdf-schema/.

John Carroll, Guido Minnen, and Ted Briscoe, forthcoming. Parser Evaluation Using a Grammatical Relation Annotation Scheme. In Anne Abeill (ed.) *Treebanks: Building and Using Syntactically Annotated Corpora,* Kluwer Academic Publishers.

Eugene Charniak, 1996. Tree-bank grammars. *Proceedings of the 13th National Conference on Artificial Intelligence*, AAAI 96, 1031-36.

James Clark (ed.), 1999. XSL Transformations (XSLT). http://www.w3.org/TR/xslt.

James Clark and Steven DeRose, 1999. XML Path Language. http://www.w3.org/TR/xpath.

Philip Harrison, Steven Abney, Ezra Black, Dan Flickinger, Claudia Gdaniec, Ralph Grishman, Don Hindle, Bob Ingria, Mitch Marcus, Beatrice Santorini, and Tomek Strzalkowski, 1991. Evaluating syntax performance of parser/grammars of English. *Proceedings of the Workshop on Evaluating Natural Language Processing Systems,* 71-77.

Nancy Ide, Patrice Bonhomme, and Laurent Romary, 2000. XCES: An XML-based Standard for Linguistic Corpora. *Proceedings of the Second Language Resources and Evaluation Conference (LREC),* 825-30.

Nancy Ide, Adam Kilgarriff, and Laurent Romary, 2000. A Formal Model of Dictionary Structure and Content. In *Proceedings of EURALEX'00*, 113-126.

Ora Lassila and Ralph Swick, 1999. Resource Description framework (RDF) Model and Syntax. http://www.w3.org/TR/REC-rdf-syntax.

Geoffrey Leech, R. Barnett, and P. Kahrel, 1996. EAGLES *Recommendations for the Syntactic Annotation of Corpora.*

Daniel Sleator and Davy Temperley, 1993. Parsing English with a link grammar. *Third International Workshop on Parsing Technologies*.

Bangalore Srinivas, Christy Doran, Beth-Ann Hockey and Avarind Joshi, 1996. An approach to robust partial parsing and evaluation metrics. *Proceedings of the ESSLI 96 Workshop on Robust Parsing,* 70-82.

Pasi Tapanainen and Timo J rvinen. 1997. A non-projective dependency parser. *Proceedings of the 5th Conference on Applied Natural Language Processing* (ANLP'97), 64-71.

Henry Thompson, David Beech, Murray Maloney, and Noah Mendelsohn, 2000. XML Schema Part 1: Structures. http://www.w3.org/TR/xmlschema-1/.